A SIMULATION MODEL FOR THE STUDY OF JOB SCHEDULING POLICY

Erik Fiegl

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

A Simulation Model for the
Study of Job Scheduling Policy

by

Erik Fiegl

June 1977

Thesis Advisor: N. F. Schneidewind

Approved for public release; distribution unlimited.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Simulation Model for the Study of Job Scheduling Policy		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1977
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Erik Fiegl		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, California 93940		12. REPORT DATE June 1977
		13. NUMBER OF PAGES 183
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Naval Postgraduate School Monterey, California 93940		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Simulation Job Scheduling

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A job scheduling simulation model was produced for the purpose of developing an effective Initiator usage policy. Whie the first part of the study gives a detailed overview of the structure of the OS/MVT Job Management routines the parts thereafter are devoted to describing the simulation model and to an analysis of Naval Postgraduate School Computer

Center operational data. A user's manual, a demonstration run with results, and a program listing of the simulation model are included.

DD Form 1473
1 Jan 73
S/N 0102-014-6601

A SIMULATION MODEL FOR THE STUDY OF JOB SCHEDULING POLICY

by

Erik Fiegl
Kapitaenleutnant, Federal German Navy

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL

June 1977

ABSTRACT

A job scheduling simulation model was produced for the purpose of developing an effective Initiator usage policy. While the first part of the study gives a detailed overview of the structure of the OS/MVT Job Management routines the parts thereafter are devoted to describing the simulation model and to an analysis of Naval Postgraduate School Computer Center operational data. A user's manual, a demonstration run with results, and a program listing of the simulation model are included.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

I. INTRODUCTION

The term 'job scheduling' describes the process of deciding the order in which a set of jobs is to be carried out. Within the data processing literature this term is used in three different ways.

In early computer systems, which were controlled by a batch monitor, there was only one job at a time in the computer. The operator had to decide which job would run next. In this connection the term 'job scheduling' was used to describe the manual operation of preparing, selecting, and feeding jobs into the system.

With the development of spooling techniques and multi-programming systems the manual scheduling operation was automated. Now jobs are fed into the system and spooled. Special job scheduling routines, which are part of the operating system, determine which of the jobs in the queue can be started next. The decision is based upon criteria such as availability of resources and job priority. This kind of 'job scheduling' is sometimes referred to as 'high-level scheduling'.

Once a job is started in a multi-programming or a time-sharing system, it is run concurrently with other jobs. The determination of which job gets the CPU next is the third area where the term 'job scheduling' or 'low-level scheduling' is used.

For this thesis the second definition of 'job scheduling' is used.

The IBM operating system OS/MVT was studied for the following reasons:

1) This system is used in the W. R. Church Computer Center at the Naval Postgraduate School and detailed literature about its structure was available.

2) The IBM OS/MVT is a general purpose operating system designed to handle a wide variety of applications. Its Job Mangement routines cover most of the aspects connected with job scheduling.

3) In addition, OS/MVT has one interesting feature: Although it is highly automated, certain job scheduling functions (starting Initiators and assigning job classes) still require operator interaction. Immediately some questions arise: Is there an optimal Initiator strategy? Is it possible to support or even replace the manual functions by an automated process?

While the first part of this thesis gives a detailed overview of the structure of the OS/MVT Job Management routines, the parts thereafter are devoted to describing a simulation model and to an analysis of operational data at NPS. The model and the evaluated data could be used to test different Initiator policies.

II. IBM OS/MVT

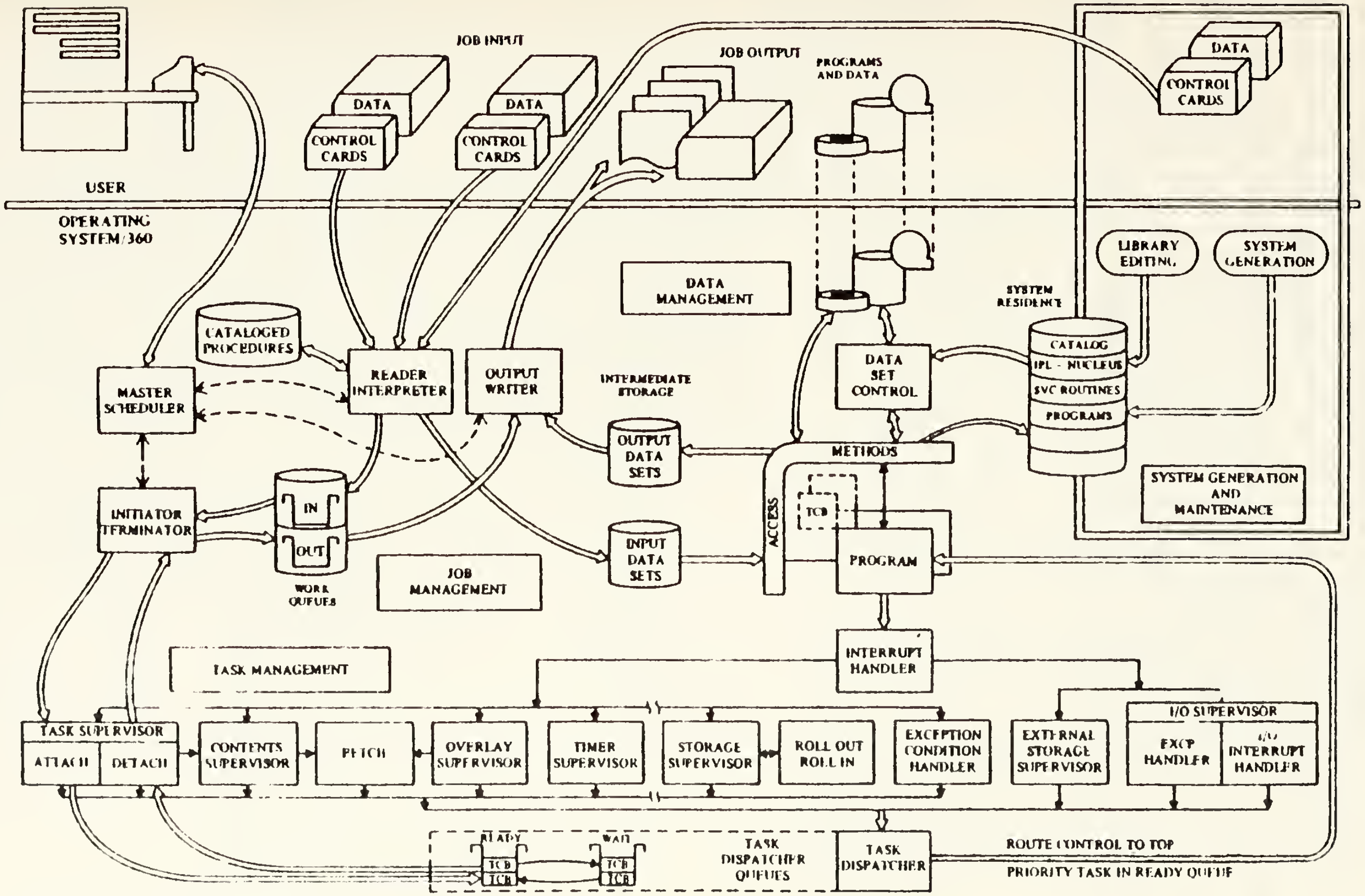
A. SYSTEM OVERVIEW

The IBM System/360 Operating System (OS) is a general purpose operating system which exists in three different versions: the Primary Control Program (PCP), Multiprogamming with a Fixed Number of Tasks (MFT), and Multiprogramming with a Variable Number of Tasks (MVT).

OS/PCP is intended for small systems and provides only basic functions and sequential job scheduling.

In addition to the basic functions, OS/MFT supports multiple input readers and output writers as well as graphics and telecommunication devices. It also has a time slicing feature and allows concurrent execution of up to 15 programs in fixed partitions.

OS/MVT, which is of special interest for this thesis, includes all facilities of OS/MFT but allows the partition (region) size to vary dynamically depending upon the needs of the particular program. In addition, it has subtasking, roll-in/roll-out, multiprocessing capabilities and a time-sharing option (TSO). Both OS/MFT and OS/MVT provide priority scheduling where programs are multiprogrammed within each priority class. A structural overview is given in Figure 1.



Source: IBM OS/360 Chart

Figure 1 - STRUCTURE OF OS/MVT

OS/MVT consists of two main parts: a Control Program and a set of Processing Programs.

The Control Program is the heart of the operating system. It is designed to manage the overall operation of the computing system and to allocate system resources in order to satisfy user requirements as well as system needs. It consists of several routines which can be grouped into five classes: Task Management, Job Management, Data Management, Volume Management, and Recovery Management.

Task Management routines, often referred as the Supervisor, control the execution of all work done in the computing system. They also serve as an interface between hardware and software. In general, the following functions are performed:

* handling interrupts
* supervising tasks
* controlling programs in main storage
* controlling main storage itself
* supervising the timer
* maintaining the system log

Job Management routines serve as a communication interface between the Control Program on one side and the operator and the users on the other. This communication processing is divided into the following functions:

* reading, scheduling, and executing operator commands
* reading the input job stream
* analysing the Job Cotrol Language
* initiating jobs for execution
* obtaining system resources
* processing the termination of jobs
* writing system messages and system output

Data Management routines handle the interface between programs and auxiliary storage. This includes:

* performing data access functions
* performing input/output support functions (OPEN, CLOSE)
* managing input/output buffers
* maintaining the data set catalog
* supplying program library facilities

Volume Management routines are used to check the condition of tapes and tape drives. They monitor the number of read and write errors for a given volume and provide error statistics and analysis. For example, a rapidly rising rate of errors would indicate the probability of a deteriorating tape and actions to rescue its contents could be taken.

Recovery Management routines try to recover from CPU and I/O hardware failures and record critical machine and program data in case of machine malfunctions.

Processing Programs , which normally run under the supervision of the Control Program, fall into three categories:

* language processors (ASSEMBLER, FORTRAN Compiler PL/1 Compiler, etc.)
* service programs (Linkage Editor, Loader, System Utilities, etc.)
* user programs

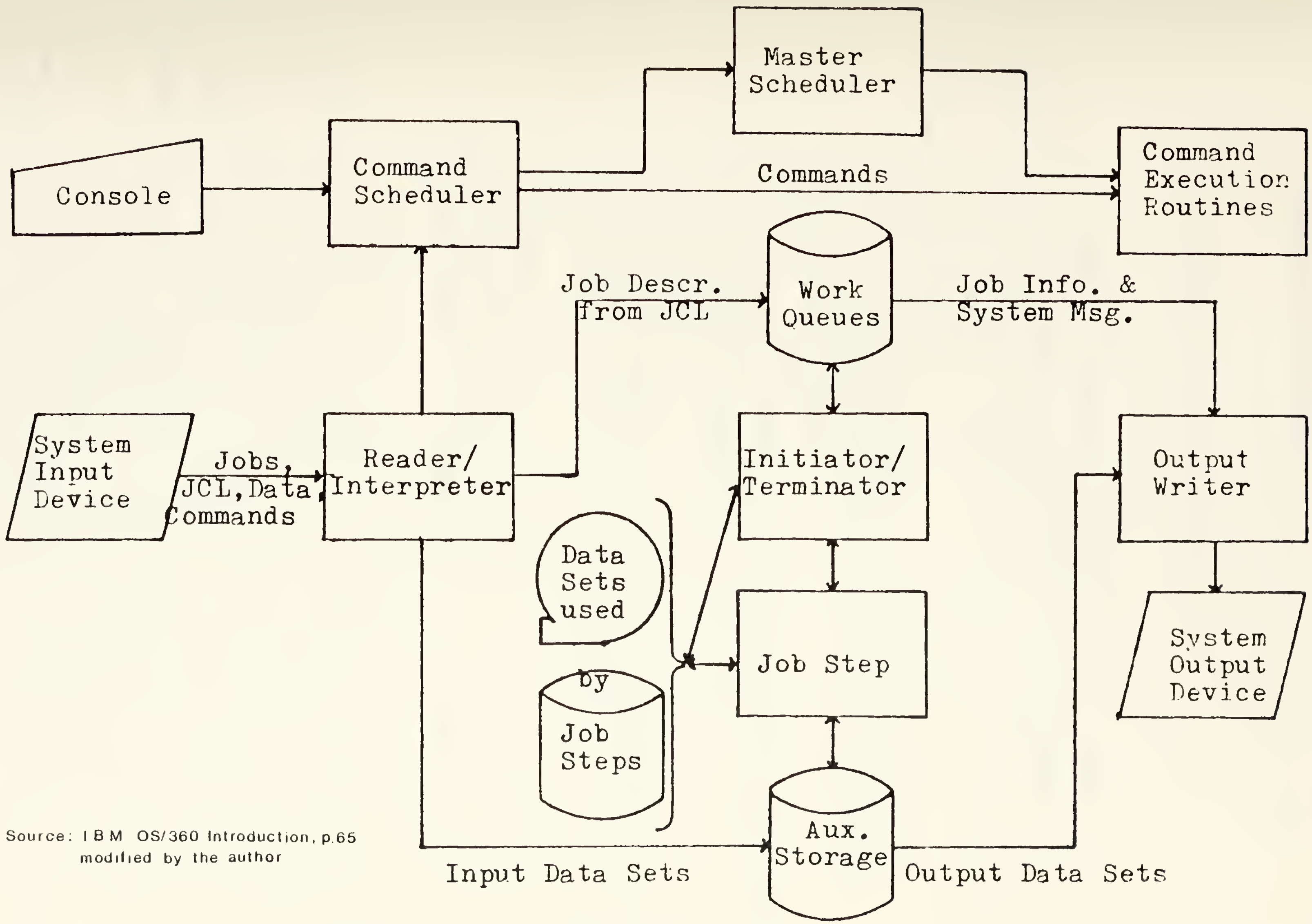
B. JOB MANAGEMENT ROUTINES

Job Management routines can be separated into two categories: command processing and job processing routines.

The command processing routines are the first system tasks which are established during Initial Program Load. They provide an interface between the system and the operator by handling operator commands and writing out system messages to the operator.

The job processing routines handle the input of user jobs, their scheduling, and their termination and exit from the system. This involves three kinds of tasks: Reader/Interpreter, Initiator/Terminator, and Output Writer. All these tasks are created by the operator with certain commands, such as START INIT.id,,,class . The operator may also modify or delete some of these tasks later. Because OS/MVT is a multiprogramming operating system all Job Management routines are executed separately from and concurrently with each other and other system or user tasks.

An overview of the data flow within the Job Management routines is given in Figure 2.



Source: IBM OS/360 Introduction, p.65
modified by the author

Figure 2 - JOB MANAGEMENT: DATA FLOW

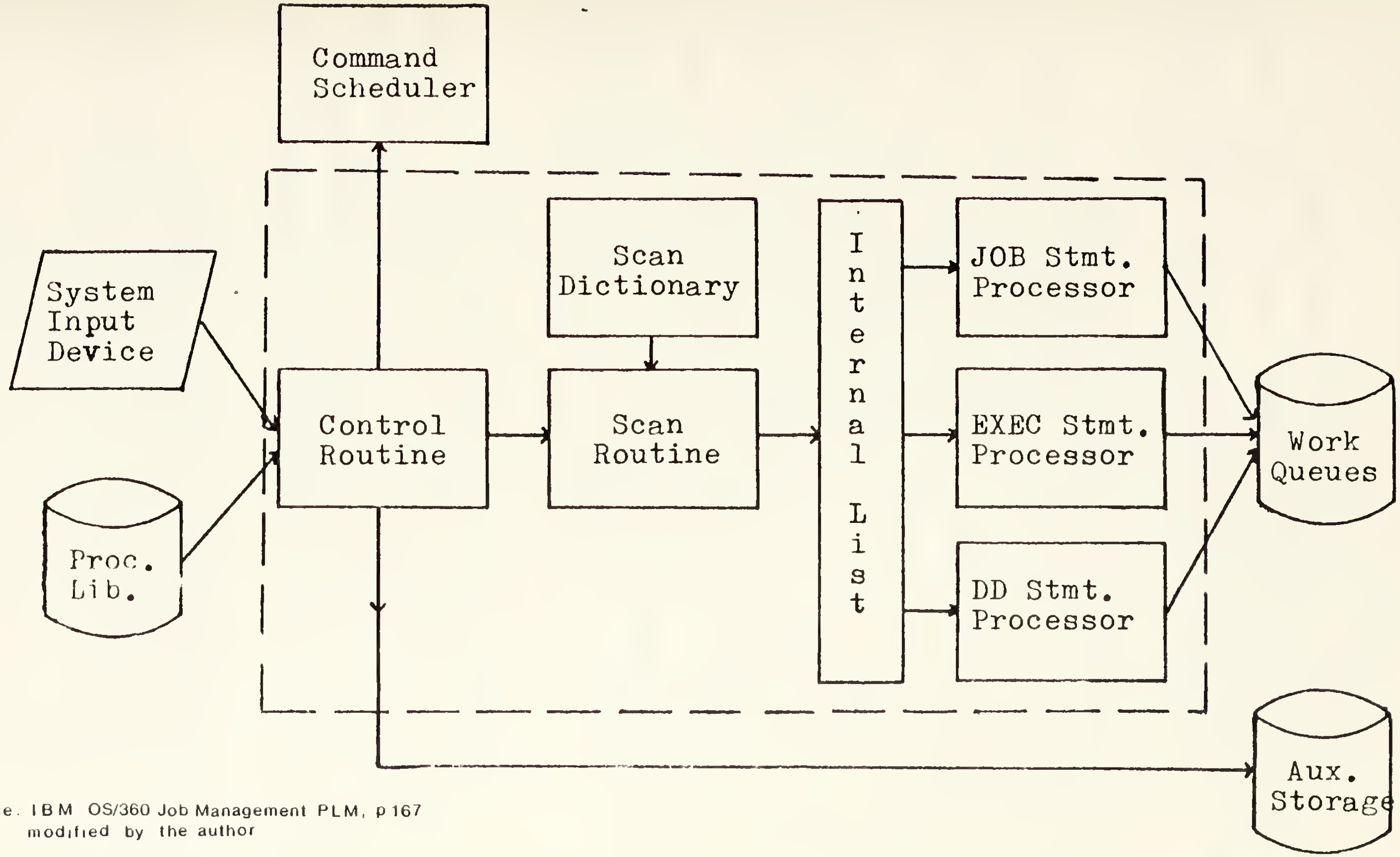
1. Reader/Interpreter

a. Functional Description

The Reader/Interpreter (Figure 3) is responsible for handling the input job stream. It reads it into the system and prepares it for further processing by other system tasks. More specifically, it performs the following functions:

* reads input job stream and Procedure Library
* scans and interprets JCL statements and builds appropriate tables
* creates output queue entries for output data sets
* places system messages to the user into the output queue
* writes input data to auxiliary storage and places the appropriate pointer to it into the job input queue entry
* enqueues the job input queue entry according to job class and priority
* passes operator commands to the command scheduler

Described above is a combined Reader/Interpreter which operates as a single task. Depending upon the needs of a given installation it is possible to split the functions into separate reading and interpreting tasks. The Reader/Interpreter is started by the operator via a START Reader command. Since more then one input device is allowed in an OS/MVT system it is also possible to start more then one Reader/Interpreter. The performance of the Reader/Interpreter is terminated either when the operator issues a STOP command or when the associated input device signals that the input job stream is exhausted.



Source. IBM OS/360 Job Management PLM, p 167
modified by the author

Figure 3 - READER/INTERPRETER: DATA FLOW

b. Input Job Stream

User jobs are given to the system in the form of an input job stream. This consists of Job Control Language (JCL), input data, and optional operator commands. A typical input job stream is shown in Figure 4.

The JCL provides a description of the job and its resource requirements. The JOB card informs the system about the job itself, such as job name, priority, job class, accounting information, etc. Each job consists of one or more steps. These are defined by the programmer and arranged in the order in which they should be processed. Each step is identified by an EXEC card which gives information about the program to be executed. This program could be system-supplied, like a compiler or the Loader, or it could be be a program previously created by the user. Any data set which is accessed or created by a job step must be defined on a DD card. Necessary information are data set name, device, storage size, etc.

The input data are records which are stored on auxiliary storage and later passed to those programs described in a job step. For example, this could be code to be translated by a compiler as well as some numerical data to be processed by a previously created user program.

Certain operator commands could also be part of the input job stream. They are separated and passed on to the command scheduler for further processing.

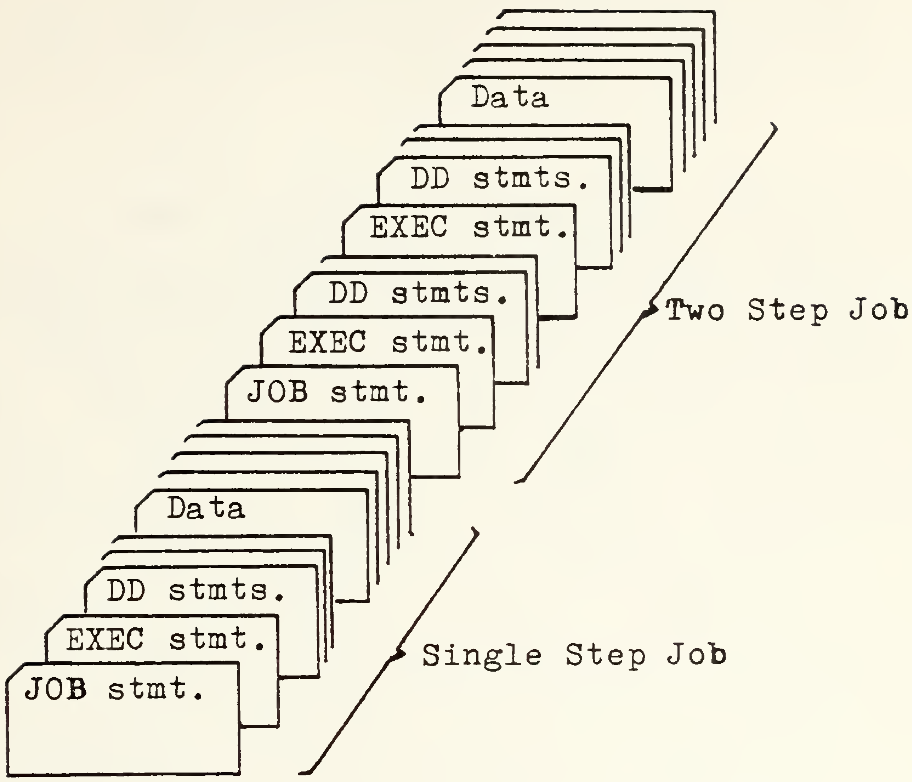


Figure 4 - INPUT JOB STREAM

c. Input Queues

The input queues belong to the work queues. They are temporary storage areas which allow work to be stored according to the input sequence, but to be processed in some user- or operator-defined priority sequence. Not only the Reader/Interpreter, but also the Initiator/Terminator and other system tasks have access to the work queues.

In the OS/MVT there are 76 work queues, 15 of which are input queues and 36 are output queues. They are maintained by a set of routines common to all Job Management tasks.

When the Reader/Interpreter processes the input job stream it translates the JCL into a series of tables [Job Control Table (JCT), Step Control Table (SCT), Step I/O Table (SIOT), and Job File Control Block (JFCB)]. These tables - ecxept those describing an output data set - form an input queue entry. This entry is enqueued when the interpretation of one job is completed. In standard IBM JCL usage, it is placed into that queue defined by the CLASS parameter on the JOB card. Possible classes are A through O. Also under standard JCL usage, the position within the queue is in accordance with the priority specified in the PRTY parameter on the JOB card; for equal priorities it is in the sequence of arrival. If no class and/or priority is specified a default value is assumed by the system, i.e. CLASS=A and PRTY= some value defined at system generation time.

In installations which do not use the CLASS parameter (Naval Postgraduate School) class is determined by user specified resources, e.g. CPU time and core usage, or the appropriate default values. Similarly, for installations which do not use the PRTY parameter (Naval Postgraduate School), position in the queue within class is determined either by a system computed priority or by order of arrival.

2. Initiator/Terminator

a. Functional Description

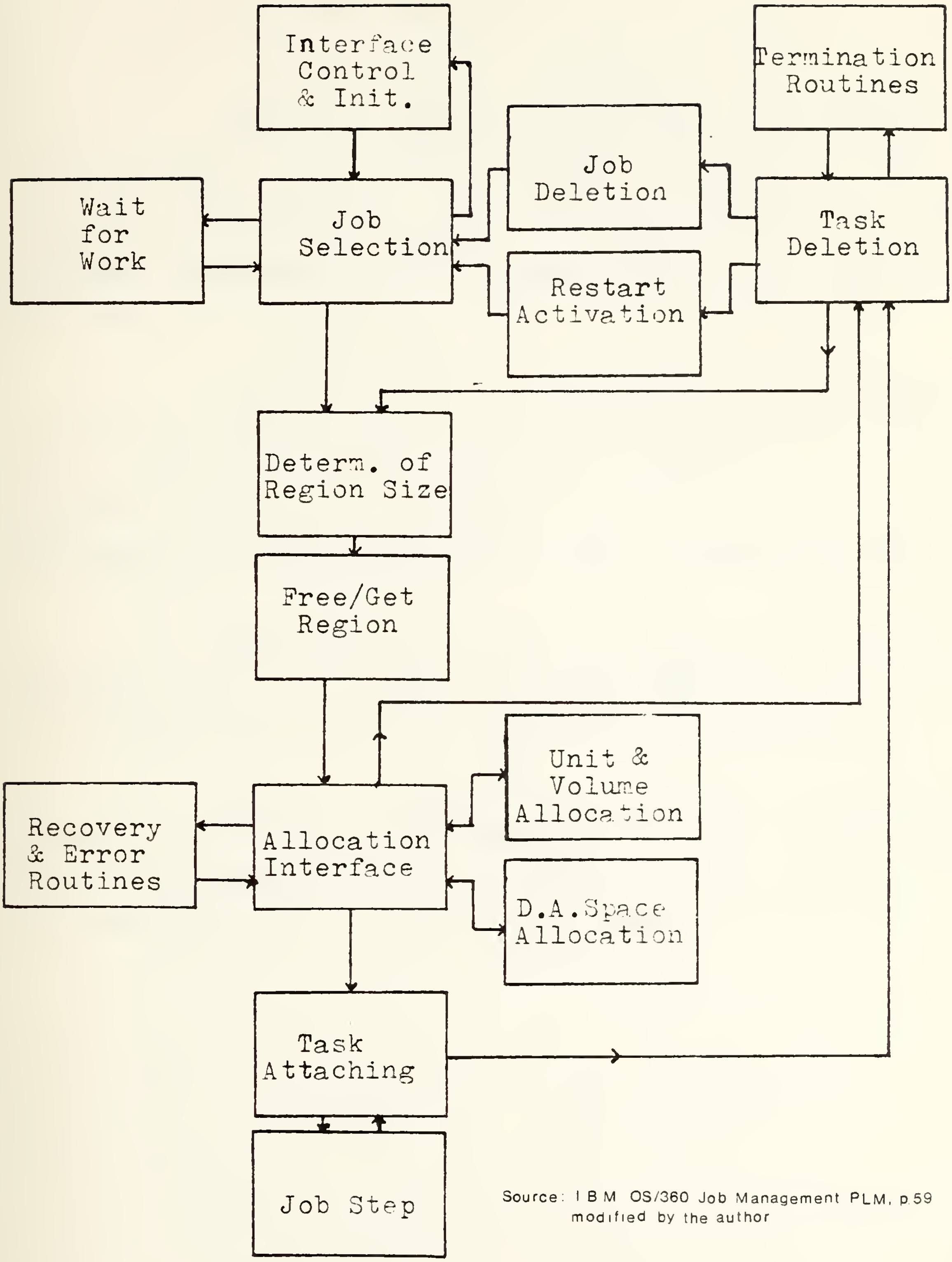
Regardless of how many jobs are read into the system by the Reader/Interpreter no job can be started without the existence of an Initiator/Terminator task. This task is created by the operator with a START command specifying an Initiator and its assigned job classes. An overview of the flow of control within an Initiator/Terminator is given in Figure 5.

The initiating part of the Initiator/Terminator selects jobs and prepares job steps for execution. In particular, the following functions are performed:

* job selection
* region management
* I/O device allocation
* task attaching

When a job step is complete the termination routines of the Initiator/Terminator release I/O devices and dispose data sets used by this step. They also update appropriate system tables. Upon completion of the last step of a job some additional system bookkeeping is done.

One Initiator/Terminator processes only one job (and within a job only one step) at a time. Multiprogramming is achieved when more than one Initiator/Terminator is started by the operator. Thus, the number of job steps that can be executed concurrently (i.e. the degree of multiprogramming) is equal to the number of active Initiator/Terminators in the system.



Source: I B M OS/360 Job Management PLM, p.59
modified by the author

Figure 5 - INITIATOR/TERMINATOR: FLOW OF CONTROL

b. Job Selection

When an Initiator/Terminator is started the operand field of the START command contains a list of assigned job classes. The order in which these classes appear determines the order in which the job selection routine searches the job input queues for jobs to be started.

If there are no jobs available in any of the associated classes, then the Initiator/Terminator enters a wait state and sends a message 'Waiting for work' to the operator. This state is kept until an appropriate job arrives or the operator issues a STOP or MODIFY command.

When a job is found the Initiator/Terminator dequeues its entry from the job input queue and passes it to the region management routines for further processing.

Figure 6 gives some examples of Initiators performing job selection or waiting for work. This figure also demonstrates the interaction of Reader/Interpreter and Initiator/Terminator with the input queues.

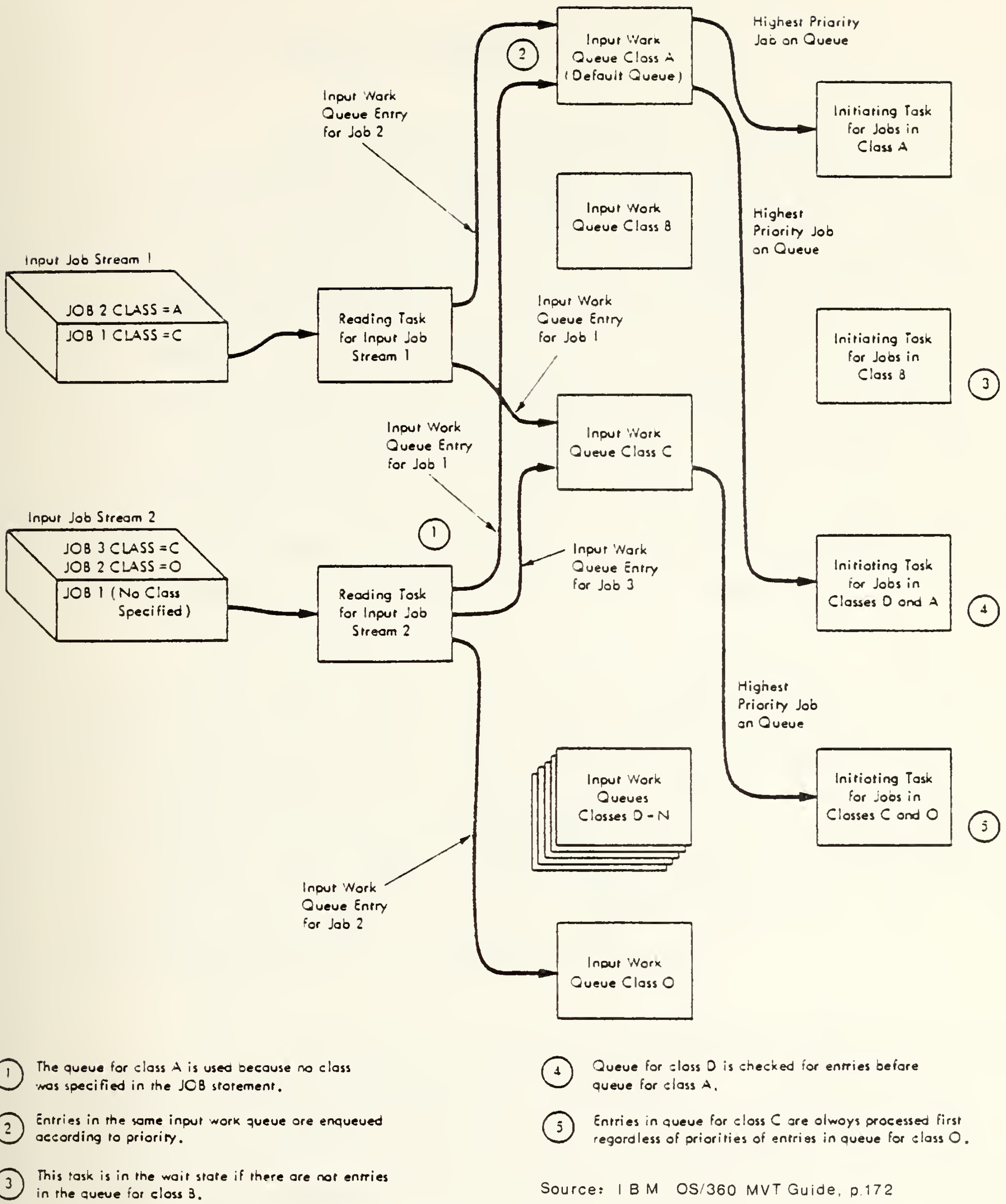


Figure 6 - JOB SELECTION

c. Region Management

The region management routines of the Initiator/Terminator dertermine the requirements for main storage of the current job step. They free the region now being used and request a new region using the FREE MAIN and GET MAIN system macros. The new region size is the larger of either the size required by the job step or the minimum core size needed by the Initiator/Terminator itself. If there is not enough contiguous core in the Dynamic Area the initializing task is put into a wait state until another task releases some core and enough main storage becomes available. When the Initiator/Terminator must wait for work the currently used region is released completely.

Normally the Initiator/Terminator uses the region of the last job step terminated. But at the very first step or at each first step after waiting for work such a region does not exist. For these cases some small routines of the Initiator/Terminator reside permanently in the Link Pack Area. They are capable of requesting a minimum core size in the Dynamic Area to get other Initiator/Terminator routines started.

Space for a region is assigned from the Dynamic Area of the main storage (see Figure 7). The assignment is made in contiguous blocks of 2 K bytes beginning from the highest available address in the Dynamic Area. The smallest region size needed by an Initiator/Terminator is 12 K bytes. To improve system performance a minimum region size of 52 K bytes is recommended by IBM.

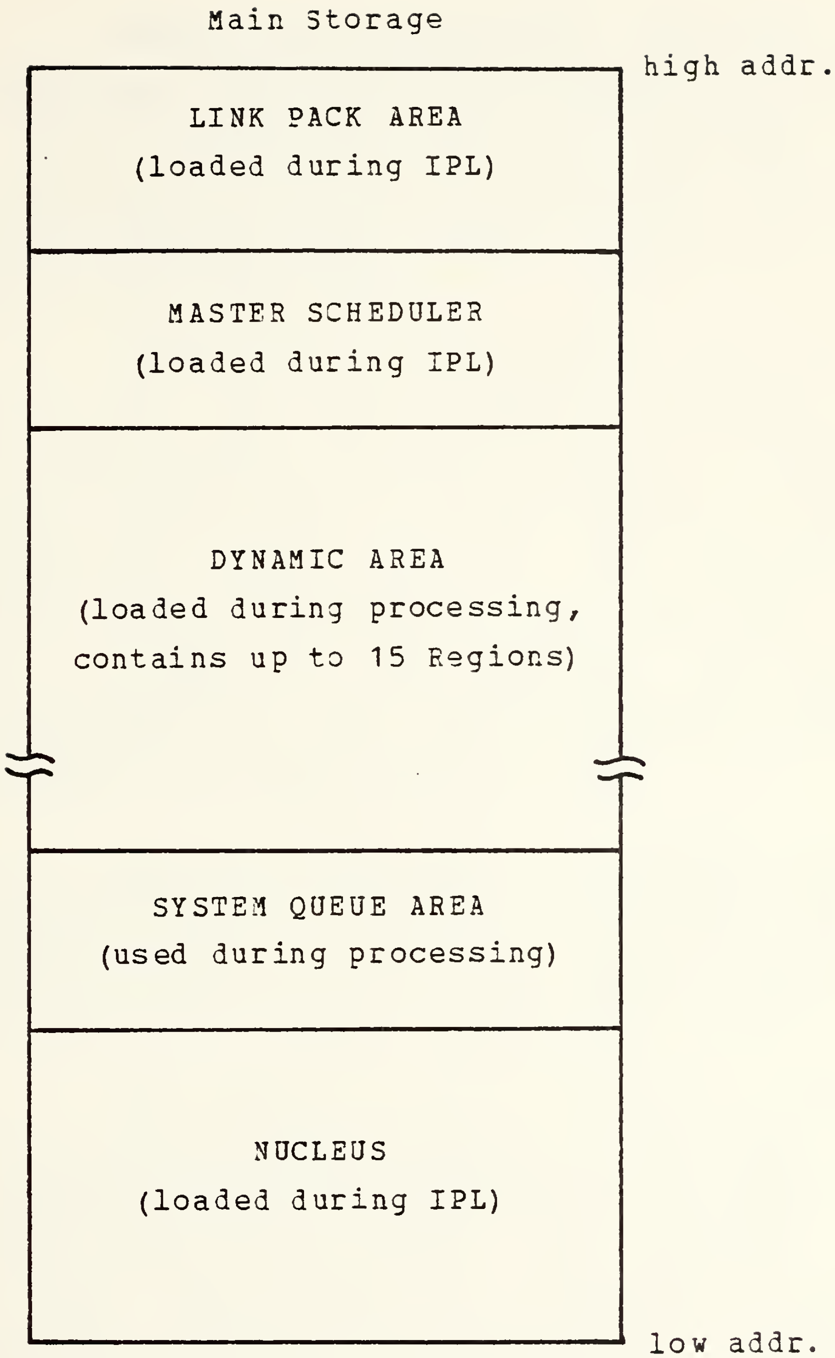


Figure 7 - UTILIZATION OF MAIN STORAGE

d. I/O Device Allocation

The I/O device allocation routines handle the I/O requirements specified in the DD statement for each job step. If some requirements are not specified completely then information gathering routines search catalogs, check status of devices, or use default values to fill these information gaps implicitly.

The input data sets used by a step are located and the device allocation routines determine if any I/O devices are available for these data sets. A step cannot be initiated unless there are enough devices - for both input and output data sets. If there are sufficient devices available they are allocated to that job step. Otherwise a message is issued to the operator. He may put the Initiator/Terminator into a wait state until enough devices are available, or he may cancel the job.

After all devices are allocated the Task Input Output Table (TIOT) is built. It contains pointers and necessary information for other system routines to allow processing of each data set used by the job step.

For volumes which are not yet mounted mount messages are issued to the operator. For output data sets which require direct access space the amount of space is calculated and checked against the available space. If enough space is available it is assigned to that job step, otherwise the allocation recovery routine is entered. If there is another task active using temporary direct access space, then this routine puts the Initiator/Terminator into a wait state until this space is released. Otherwise the recovery routine informs the operator. He may put the

Initiator/Terminator into a wait state until he can make some space available, or he may cancel the job.

Before the I/O device allocation routines can pass control to the task attaching routines, they have to wait until all volumes, for which mounting requests have been issued, are mounted. A final check is made to verify that the mounted volumes are correct. If an incorrect volume has been mounted, a demount instruction followed by a new mount message is issued to the operator. The I/O device allocation routines again have to wait until this error is corrected.

e. Task Attaching

When all resource requirements of a job step can be satisfied the final operation prior to starting this step is to attach the user task to the Supervisor. For this purpose the attaching routines gather information (dispatching priority, remaining job run time, etc.) needed by the Supervisor. This information is placed into a Task Control Block (TCB) and queued into the TCB queue.

The initiation of the job step is now complete and the step will run under the control of the Supervisor. The Initiator/Terminator task is placed into the wait state until this step is to be terminated.

f. Step and Job Termination

A job step is terminated either normally when it is complete or abnormally when an error prevents further processing, when a specified time interval has expired, or when the job is cancelled by the operator. In any case the Supervisor activates that Initiator/Terminator which started this job step. The termination process is then done by this Initiatior/Terminator using the region of the just ended step.

The termination routines direct the disposition of data sets and the release of I/O devices used by the job step. They also update appropriate system tables.

If there are more steps control is passed to the region management routines to initiate the next step.

If the last step of a job is terminated some additional processing must be done. The job entry from the input queue is removed completely and entries of the jobs's system output data sets are enqueued into the appropriate output work queues. Control is then passed to the job selection routines of the Initiator/Terminator.

3. Output Writer

a. Functional Description

In a multiprogramming system a job is usually not allowed to use a printer or a punch directly, even if it is the only job in the system at that time. Hence, a job generally writes on intermediate output data sets on direct access devices. When a job terminates, pointers to those data sets as well as system messages concerning that jcb are enqueued into output queues. Such messages and data sets are then processed by a System Output Writer.

An Output Writer is created by the operator as a result of a START Writer command. One parameter of the START command specifies the associated output device (printer, punch, or tape), another parameter describes a single output class or a group of up to eight classes. More then one Output Writer may be started, depending upon the needs of an installation.

An Output Writer controls the writing of all system output within its specified class(es). It enqueues entries from the output queues and performs the required output operation on its assigned system output device. When all entries of the assigned class(es) have been processed, the Output Writer is placed into a 'wait' state. It is again made ready when a job terminates and the Initiator/Terminator places an entry into a queue associated with this Output Writer.

b. System Output and Output Classes

The system output consists of messages from the operating system to the programmer and of data sets created by the job and designated by the programmer in a DD statement. The messages and pointers to the data sets are placed as entries into the system output queues by the Reader/Interpreter and by the Initiator/Terminator.

The system output is divided into 36 classes and there is one output queue corresponding to each class. The classes are named with single letters (A-Z) or digits (0-9). The names have no inherant meaning but are simply used to group output of similar characteristics. There might be, for example, one class for output to a printer, another class might be for punched output, and a third class might be for output written on tape for later printing.

c. Direct System Output

By use of Direct System Output Writers there is a possibility that jobs can directly use system output devices such as printer, punch, or tape. Direct System Output Writers are started by the operator and assigned to a certain device and to specific output classes. The assignment to a job is made by the I/O device allocation routines of an Initiator/Terminator. The selected writer is then tied to that job for the duration of all job steps. When the job writes its output, the output will go directly to the specified device. In addition, this job may also produce output of other output classes which will be spooled and later processed by normal Output Writers as described earlier.

C. REMARKS

As described earlier an Initiator/Terminator serves only one job at a time. This means that, once the Initiator/Terminator routines have selected a job, they are tied to it until this job has terminated. This might be a good approach, since it establishes a well defined relationship between an Initiator/Terminator and a user's job. However, there is an obvious disadvantage which shows up whenever an Initiator/Terminator has to wait for a requested resource. When waiting for data sets to be mounted or for other operator interactions, this waiting time could extend from a few seconds to several minutes. During that time the Initiator/Terminator remains dormant. Since it is tied to one job only it cannot serve any other job. This means that during waiting periods the degree of multiprogramming is decreased by one.

OS/MVT allows and even requires a high amount of operator interaction. At Initial Program Load the operator has to set system parameters and he must start and assign classes to Readers, Writers, and Initiators. Later he may stop, modify or start new system tasks, or he can hold, reset, or cancel certain jobs and can connect and disconnect specific devices. According to IBM this provides a great amount of flexibility. At any time the performance of certain system tasks could be tailored to the need of a given job load. The ability to supervise large parts of the operating system allows the operator to handle even very extreme cases and emergency situations which could not be handled by a fully automated system.

The disadvantage of this approach is that the performance of this sophisticated and otherwise highly automated operating system is mainly influenced by the experience and performance of the operator.

The job scheduling algorithm used in OS/MVT requires that all resources be pre-allocated to a job step. That means , no job step can be started until sufficient main memory and all requested data sets and devices are available. The lack of only one of the needed resources will cause - depending upon the circumstances - either a wait until this resource becomes available , or a cancellation of the job step, or an intervention request to the operator.

The purpose for this kind of resource control is to avoid deadlocks which could arise from simultaneous use of the same resources by several tasks. This strategy may be sometimes inefficient and costly since some of the resources allocated to a job step remain unused for a long period of time or they may not even be used at all. Practice has shown that the possibility of a deadlock between several tasks initiated by one or more job steps has not been eliminated completely. There is a possibility that the use of ENQ and DEQ macros to control resource allocation may create a 'circular wait'.

III. SIMULATION MODEL

A. OVERVIEW

The model simulates the main functions of the IBM OS/MVT Job Management routines. In general, the overall structure of the operating system is reflected in the structure of the model, but since OS/MVT is a very complex system some simplifications and limitations are necessary. They are described in the following parts.

The purpose of the model is to test different Initiator strategies under certain job loads and operating conditions. To achieve this goal the number of Initiators (up to 15) and their associated job classes (up to 8 per Initiator) can be varied during the simulation. The model also allows important system parameters (size of main memory, input spool space, number of I/O devices, etc.) to be entered. By varying these parameters the simulation program can be tailored to a certain extent to a given environment.

The job stream used during the simulation is generated by a job generating module. This module can be modified to allow generation of job streams with different characteristics. Some statistical routines collect and print statistical and performance data upon user request.

A special problem is the simulation of the time used by Job Management routines, by other system tasks, and by the different user jobs. The basic time measurement in the model is elapsed step run time. This is the wall clock time counted from the beginning of step initiation to the end of step termination. The elapsed job run time is the sum of all elapsed step run times of a job. Included in this time is the CPU time used by the job, the time waiting for I/O , as well as the time used by the Job Management routines and other system tasks. Since the elapsed step run time has a range of one second to several minutes, one second is used as the basic time unit in the simulation model.

The programming language PL/1 was chosen for several reasons:

* it is a block-structured language
* it allows good data structuring
* it is easy to use for I/O routines
* it is well supported at the Naval Postgraduate School, where the model was developed

In addition, PL/1 allows nearly unrestricted variable names. This makes the program more readable and self-documenting.

A functional overview of the simulation model is given in Figure 8.

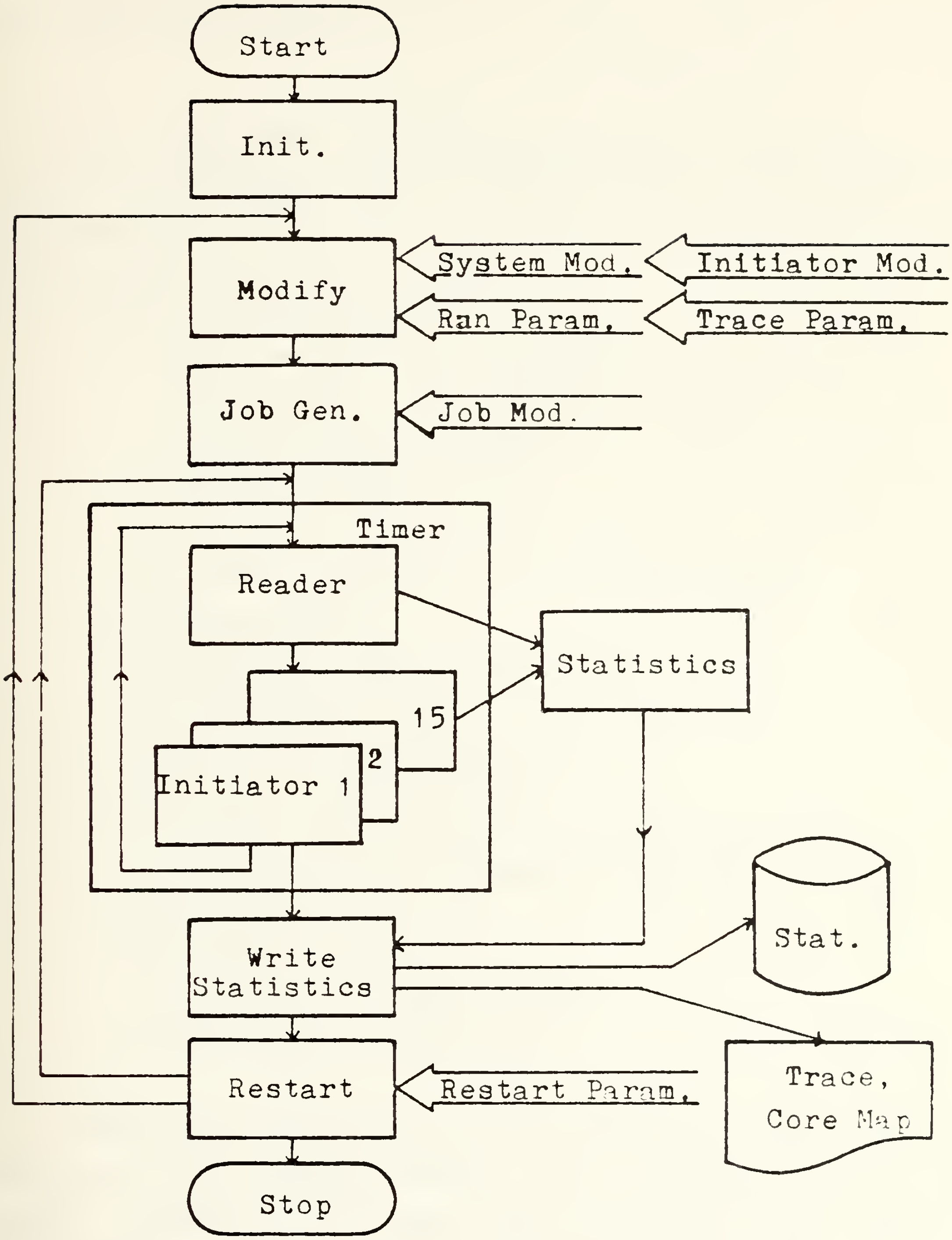


Figure 8 - STRUCTURE OF THE SIMULATION MODEL

B. SUPERVISOR MODULE

The supervisor module initializes and drives the entire simulation program. When it calls the initialization and modification routines, the user may enter the following parameters:

* system modifications:
    - main memory (high address)
    - main memory (low address)
    - number of disk drives
    - number of tape drives
    - amount of input spool space
    - amount of public direct access space
* run parameters:
    - number of jobs to be read
    - simulation time
    - job stream modifications
* Initiator modifications:
    - number of active initiators (up to 15)
    - associated job class(es) for each initiator
* trace parameters:
    - simulation trace
    - map of main memory usage
    - statistics gathering

After these parameters are entered the timer module gets control. This module checks the simulation time table, which contains the times when the Reader and each active Initiator need attention. The timer always calls the next module, which is responsible for updating the attention time. This process is terminated when the simulation time or the input job stream is exhausted, whichever comes first.

At the end of each simulation step the user has the choice to stop or restart. If restart is chosen he may run the simulation with the same or new parameters.

C. READER MODULE

It is assumed that the Reader is active during all simulation steps and that it resides in the upper part of the dynamic area in main memory. The user must note the amount of core used by the Reader when entering the main memory high address parameter.

During the initialization phase the job generating module places the requested number of jobs and their characteristics into the input job stream and also sets the time of the first job arrival into the simulation time table. When the Reader is called by the timer module it takes the next job from the input stream and enqueues it according to its class and priority into one of the job input queues. Then the Reader determines the time of next job arrival and places this time into the simulation time table as its new attention time.

If the input spool space is exhausted the reading and enqueueing of jobs is delayed until another job terminates and enough spool space becomes available. Since the supervisor ends the simulation run after the requested number of jobs has been read, the Reader will never be called when the job stream has been exhausted.

D. INITIATOR MODULE

The Initiator module simulates the functions of job selection, waiting for work, region management, device allocation, data set allocation, direct access space allocation, step termination, and job termination. All information necessary to perform these functions is maintained in an Initiator table. Since the dimension of this table is 15 it is possible to run 15 Initiators concurrently. Each Initiator updates its time of next attention in the simulation time table.

1. Job Selection and Waiting for Work

An Initiator can be associated with up to 8 different job classes. To find the next job the input queues are searched in the order in which the classes were assigned to Initiators by the user.

If there is no job of the appropriate class in the queues, the Initiator releases its region and is put into a 'wait for work' state. This state is kept until a new job arrives. Then the Initiator gets a new region of a pre-defined minimum size and the queues are searched again. If a job is found it is dequeued and associated with its Initiator for further processing.

2. Region Management

When a job is selected the region size of its first step is determined and the region currently used by the Initiator is released. The new region is allocated from the top of the Dynamic Area in main memory. If insufficient contiguous core is available, the Initiator is placed in a 'wait for core' state. It is activated again for a new region allocation when another job ends and core is released. The region management routines are called at the beginning of each job step.

It is assumed that the size of the Dynamic Area is fixed during the simulation. However, the user must set the upper and lower addresses. He can account for the size of the system queues by setting the appropriate lower address. He must also account for the amount of storage used by system tasks, by Reader(s), Writer(s) and other permanent programs by setting the appropriate upper address.

3. Device Allocation

In general, the allocation of I/O devices and I/O channels is not simulated in the model. Most devices are physically shareable (data cell, disks) or are made shareable using spooling techniques (card reader, printer, plotter). Evaluation of system logs has shown that normally all requests to such devices can be satisfied by the system immediately. The time overhead required for selection, allocation, and spooling is included in the elapsed run time of each job step.

However, this simplification is not valid in case of tape drives and disk drives with removable disk packs. The allocation of these devices sometimes requires operator interaction or causes long additional waiting times until a requested tape or disk drive becomes available.

The device allocation routines handle the tape and disk requests of each job step. If a device is not available, an operator interaction is simulated. The operator answer could be 'cancel' or 'wait'. In the first case the whole job is abended; in the second case the Initiator is put into a 'wait for device' state. Whenever another job terminates the device allocation routines are activated again until all outstanding device requests can be satisfied for the current job step. In order to avoid long waiting times, all jobs which request more devices than are installed in the system are abended.

The type of operator answer ('cancel' or 'wait') and his response time are drawn from a probability distribution.

The number of tape and disk drives can be set by the user, thus tailoring the simulation model to his needs.

4. Data Set Allocation

Only those data set allocations are of interest which require operator interaction, thus causing additional waiting time. It is assumed that for every requested tape and disk drive an appropriate volume has to be mounted. By placing the Initiator into a 'wait' state, the data set allocation routines simulate the time needed by the operator to perform the mounting.

Independant of mounting requests are verification requests. Some data sets require an operator response to verify that a user is authorized to access a data set. This case is also simulated by the data set allocation routines. Since, for the model, the operator response time is of greater interest than the reason for an operator interaction, this case could also be used to account for any additional operator request which is otherwise not covered (channel separation request, etc.).

The response time for mounting disks, mounting tapes, or answering other requests is drawn from a probability distribution as described earlier for the device allocation routines. Also the possibility of job cancellation is included in the model.

5. Direct Access Space Allocation

Only the allocation of temporary space on public direct access devices is simulated. The total amount of public space within the system can be set by the user. If a space request of a job step can not be satisfied the space allocation routine checks if there are other job steps active. If not, the current job will be abended, since its request can never be granted. Otherwise the Initiator is placed into a 'wait' state until another job ends which might release some temporary space on public direct access devices.

6. Step and Job Termination

At the end of a step all requested disks and tapes are released and given back to the system. Temporary space on public direct access devices, however, is kept until job termination. If there is another step to process, control is given to the region management routines to start the next step.

At normal job termination as well as in case of job abending all system resources (tapes, disks, public direct access space, and input spool space) are released. The Initiator table is cleared and a new job can be selected. Job termination is also posted to the Reader which might be waiting for input spool space and to other Initiators which are in a state of waiting for system resources.

E. WRITER MODULE

Spooled system output only is assumed for this simulation. That means that the Writer works independently from and concurrently with the Reader and Initiators. Since the amount of overhead due to multiprogramming with the Writer is already included in the elapsed job run times, no Writer function has to be simulated. As mentioned earlier the user, however, must deduct the core size used by the Writer from the top of the Dynamic Area in main memory.

F. STATISTICAL MODULE

Several statistical routines gather Initiator performance data. These data are maintained in a statistical table which can be written on a file upon user request. This file must then be processed and evaluated by a supporting statistical evaluation program.

As a second choice the user can request a simulation trace. Similar to the logs at the operator's console all important events (job starting, job termination, initiator waiting for work, mount requests, etc.) are printed out. As a third choice a map showing the utilization of main memory at the end of each simulation step can be printed.

Examples of a trace, a core map, and some evaluated performance data are shown in Appendix B.

IV. DATA ANALYSIS

A. SOURCE OF DATA

To drive the simulation model certain information about input job stream characteristics, system configuration, operator response times, etc. was necessary. To gather data four main sources were used:

The first source was the IBM 360/67 computer center at the Naval Postgraduate School. An overview of the hardware configuration is given in Table I; the job class definitions and the priority policy are listed in Table II. With the installation of the HASP spooling system in September/October 1976, the Quickrun class was replaced by the input class 0, which was restricted to jobs using certain cataloged procedures only, using up to 180 K of core and up to 20 seconds of CPU time. These restrictions were nearly the same as for the Quickrun class, but since some changes were made in the cataloged procedures about half of all previous class A jobs together with nearly all Quickrun jobs would now qualify for the new class 0. For the validation runs the characteristics of the Quickrun class were simulated.

The second source was data collected from the System Management Facility (SMF) routines. These routines gathered statistics about every job processed by the computer system. Contained therein were: job name, job class, job priority, job arrival time and date, job starting time and date, job

completion code, number of input cards, number of job steps, requested and used core per step, CPU time per step, elapsed time per step, sysout records per step, etc.

Only SMF data of the period from February to August 1976 were usable for the purpose of this study. Before this time period a completely different job class and priority specification was in effect. After this period some parameters used for the simulation model were no longer recorded due to the change to the HASP spooling system. So SMF tapes of April, May, and August 1976, containing data of about 75,000 jobs, were evaluated.

As a third source the complete set of system logs of August 1976 was available and used to extract certain parameters. These parameters included the number of Initiators and their associated job classes, the number of other system tasks active at the same time, and upper and lower addresses of the Dynamic Area in main memory.

Since the SMF tapes did not provide information about usage of tapes and disks the system logs were also used to count the number of tape and disk mount requests and to evaluate data such as operator mounting times, operator response times to other system requests, and the number of job cancellations by the operators.

Last, but not least, the operators themselves and other members of the computer center staff at NPS provided valuable input for the collection and evaluation of system parameters.

IBM 360/67 at NPS

NO.	UNIT	DESCRIPTION
2	2067-2	Processing Unit
1	2167-4	Configuration Control Unit
2	1052-7	Console Typewriter
2	2860-2	Selector Channel
2	2870-1	Multiplexor Channel
3	2365-12	Processor Storage (256K Bytes each)
5	MM365-12	Core Storage (256K Bytes each) - Lockheed
1	2820-1	Drum Control
1	2301-1	Drum Storage (4 M Bytes)
3	2841-1	Disk Control
8	2311-1	Disk Drives (7.25 M Bytes each)
1	2314-1	Disk Unit (8 Drives, 29 M Bytes each)
2	5314	Disk Control - Potter
16	4314	Disk Drives - Potter (29 M Bytes each)
1	2321-7	Data Cell (400 M Bytes)
1	3803-1	Tape Control
5	3420	Tape Drives
1	2803-1	Tape Control
2	2402-1	Tape Unit (2 Drives each)
1	2821-1	Control Unit
1	2821-2	Control Unit
2	1403-N1	Printer
1	2501-B2	Card Reader
1	2540-1	Card Reader/Punch
1	110	Plotter Control - CALCOMP
2	765	Plotters - CALCOMP
1	2702-1	Transmission Control Unit (30 Ports)
24	2741	Communication Terminals
8	----	Video Display Units (assorted vendors)
1	2250-1	Graphic Display Unit
1	Tek4012	Display Terminal - Tektronix
1	Tek4610	Hard-Copy Device - Tektronix
1	PIX	Paradyne PIX/Remote Job Entry
1	2701	Data Adapter Unit (with PDA)

Table I - SYSTEM HARDWARE AT NPS

EFFECTIVE 1 FEB. 1976

JOB CLASS DEFINITIONS

CLASS	REGION	TIME	TAPE/JOBSTEP
Q	QUICKRUN		none
A	180K	20s	none
B	180K	2m	≤2
C	250K	5m	≤2
D	250K	5m	>2
E	350K	5m	≤2
F	400K	30m	none
J	>400K	>30m	none
K	>400K	>30m	any

Comments:

1. Execution in each class will be on First-come First-served (FCFS) basis.
2. Classification scheme ignores SYSOUT and SYSDA requirements. Printing priority is considered separate from execution priority and is based on the actual number of lines generated.

Table II - JOB CLASS DEFINITIONS AT NPS

B. JOB STREAM CHARACTERISTICS

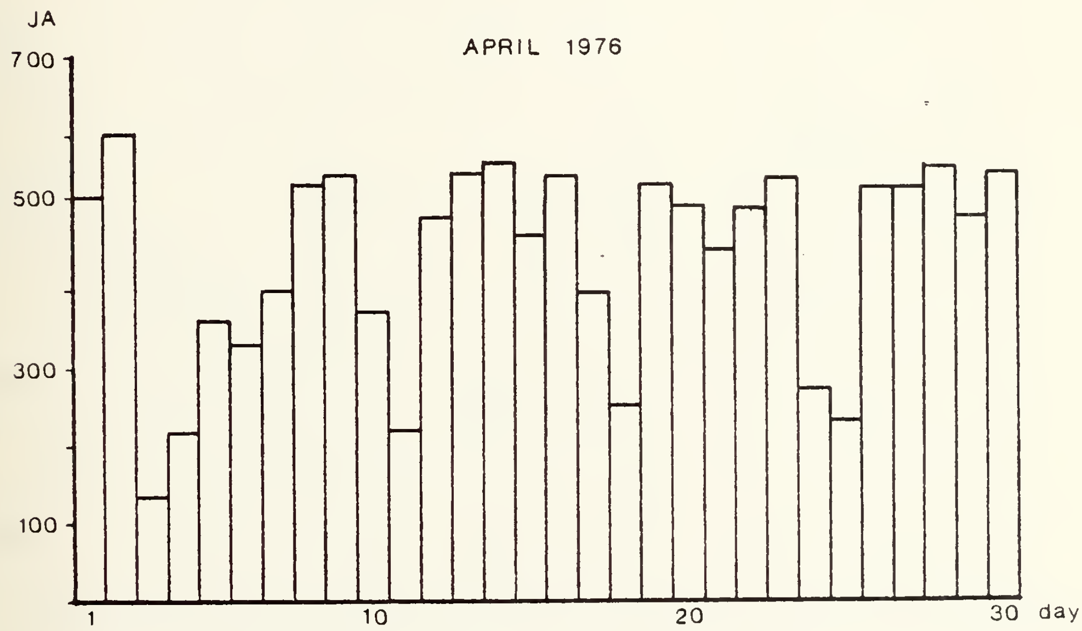
Very important for an effective simulation run were parameters which characterized the input job stream. Members of the computer center staff and students had analyzed the input job stream at the NPS computer center. But since these studies were based on jobs rather than steps, as required by the simulation model, these studies were not usable and a new evaluation had to be made.

Most of the job characteristics were extracted from the SMF tapes. When working with these tapes a few problems arose. There was no class D job observed and the number of J and K class jobs was very small. In addition some jobs in undefined job classes were present. An explanation for this was that the operators used to start one Initiator with an undefined job class. Then they reset jobs from classes J and K and the very few jobs from class D, and selected these manually for initiation. For the simulation model, classes D, J, K, and all undefined classes were collected into one class K.

Also some jobs used more core than allowed by their job class. The explanation again was that operators reset jobs from one class to another. During the evaluation these jobs were filtered out and added to job class K.

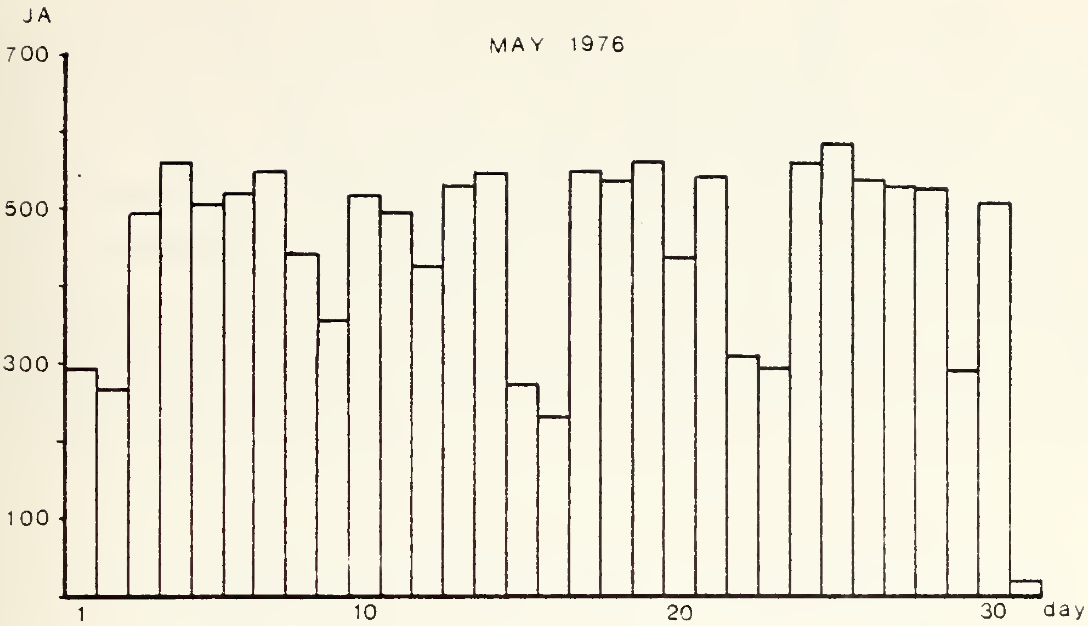
Elapsed time and core used were not recorded for the Quickrun jobs. Since these jobs had the same time and core restrictions as class A jobs (core up to 180 K Bytes, CPU time up to 20 seconds), the class A distribution was assumed.

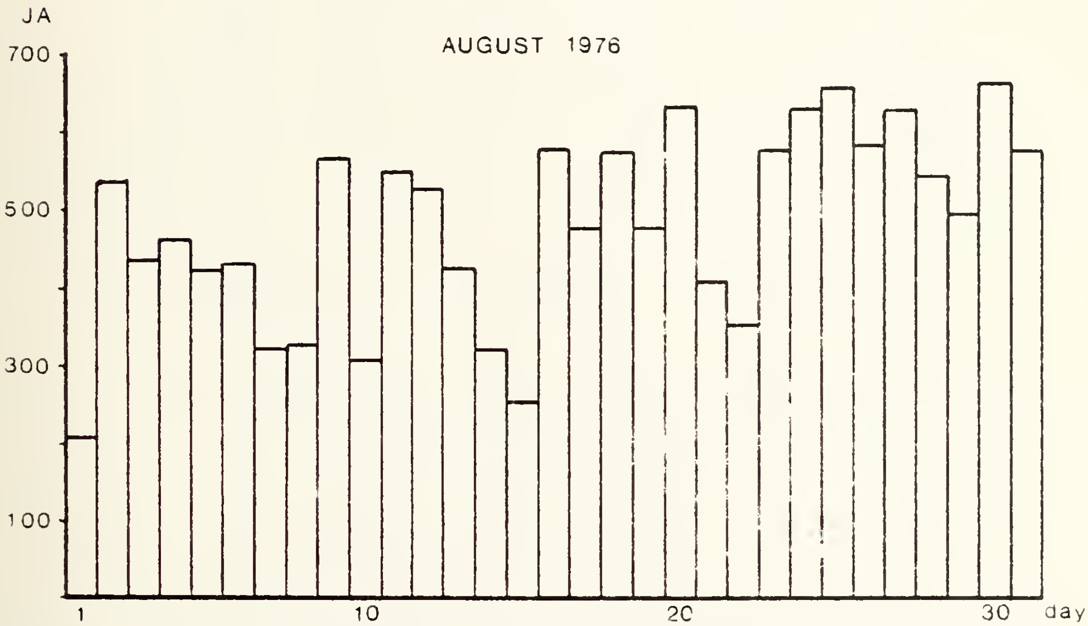
To obtain relativly stable, but still representative data the time period from 10 a.m. to 5 p.m. each day was selected. The data collection was further restricted to those days with more than 500 job arrivals within this time. Figure 9 and Figure 10 show that 47 days of the months April, May, and August 1976 met these requirements. With this approach untypical conditions which occur at night and on weekends and holidays were eliminated. Although the time of observation covered only about 15% of the total hours within the three month period, 25,532 or more than one third of all job arrivals were included.



JA : Number of job arrivals between 10 a.m. and 5 p.m.

Figure 9 - JOB ARRIVALS (APRIL 1976)





JA : Number of job arrivals between 10 a.m. and 5 p.m.

Figure 10 - JOB ARRIVALS (MAY AND AUGUST 1976)

To obtain the distribution of job arrivals a 2-hour period in each month was selected at random and the arrivals per minute were counted. As shown in Table III in each job class the distribution was very close to a Poisson distribution. In fact, the observed values easily passed a 95 percentile chi-square test to match the theoretical values. Thus for the job arrivals in the simulation model a Poisson distribution with exponentially distributed interarrival times was used. The mean job arrival rate was 1.294 jobs per minute.

	class A		class B		class C	
i	F_o	F_{th}	F_o	F_{th}	F_o	F_{th}
0	67	66.41	88	87.43	105	104.15
1	39	39.29	26	26.69	13	14.75
2	11	11.62	6	4.38	2	1.05
3	2	2.29	0	0.46	0	0.05
4	1	0.34	0	0.04	0	0.00
5	0	0.04	0	0.00	0	0.00

	class E/F		class K		class QR	
i	F_o	F_{th}	F_o	F_{th}	F_o	F_{th}
0	118	118.01	109	109.49	106	105.02
1	2	1.97	11	10.04	12	14.00
2	0	0.02	0	0.46	2	0.93
3	0	0.00	0	0.01	0	0.04

F_o : observed number of 1-min. intervals with i arrivals
F_{th} : theoretical number of 1-min. intervals with i arrivals assuming Poisson distribution

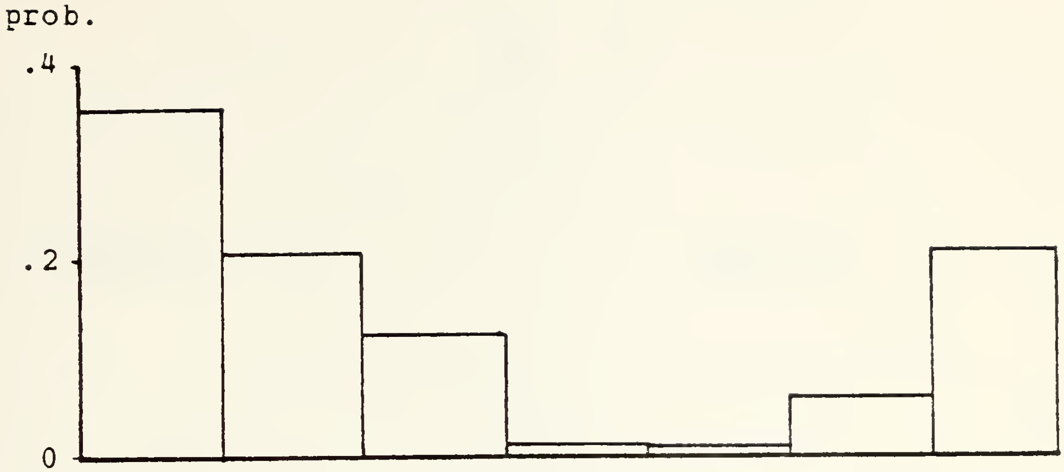
Table III - DISTRIBUTION OF JOB ARRIVALS

Other perameters evaluated from the SMF tapes were distibution cf job classes, number of steps per job, number of input cards per job, core used per step, and elapsed time per step. The probabilty distributions are given in Table IV to Table X; histograms are provided in Figure 11 to Figure 24.

It was felt that the distribution of elapsed step time might be approximated by a Gamma or possibly a Weibull distribution. Although a great amount of work was spent to match the observed values with those theoretical functions no relationship could be found.

In the simulation model the amount of public direct access space was one input parameter. The storage of system output records was only one part of this space, but other data were not available. An evaluation of the job completion codes, however, showed that within the observed time periods no job abended because of lack of public direct access space. Thus for the simulation runs no public direct access space was requested.

The number of disk and tape mount requests and the number of other system requests were evaluated from the system logs of August 1976. Only the total number of requests could be counted, but information about the associated job classes was not available. For the simulation model it was assumed that the probability of requests was the same in all job classes which qualified for the appropriate type of requests. The distribution is given in Table XI.



CLASS:	A	B	C	E	F	K	QR
PROB.:	.3664	.2168	.1254	.0114	.0110	.0515	.2175

Figure 11 - JOB CLASS DISTRIBUTION

STEPS	A	B	C	E	F	K	QR
1	.4180	.3226	.5420	.3655	.1352	.3934	.5524
2	.0531	.0878	.0615	.1586	.0569	.0845	.4476
3	.5069	.4767	.2877	.4483	.6121	.4155	.0000
4	.0086	.0367	.0285	.0103	.0107	.0350	.0000
5	.0123	.0517	.0619	.0103	.1530	.0471	.0000
6	.0006	.0216	.0172	.0035	.0249	.0152	.0000
7	.0000	.0025	.0006	.0000	.0000	.0046	.0000
8	.0000	.0002	.0000	.0000	.0036	.0008	.0000
9	.0003	.0000	.0000	.0035	.0036	.0015	.0000
10	.0001	.0002	.0003	.0000	.0000	.0000	.0000
11	.0001	.0000	.0003	.0000	.0000	.0008	.0000
12	.0000	.0000	.0000	.0000	.0000	.0008	.0000
13	.0000	.0000	.0000	.0000	.0000	.0000	.0000
14	.0000	.0000	.0000	.0000	.0000	.0000	.0000
15	.0000	.0000	.0000	.0000	.0000	.0008	.0000

Table IV - DISTRIBUTION OF JOB STEPS PER CLASS

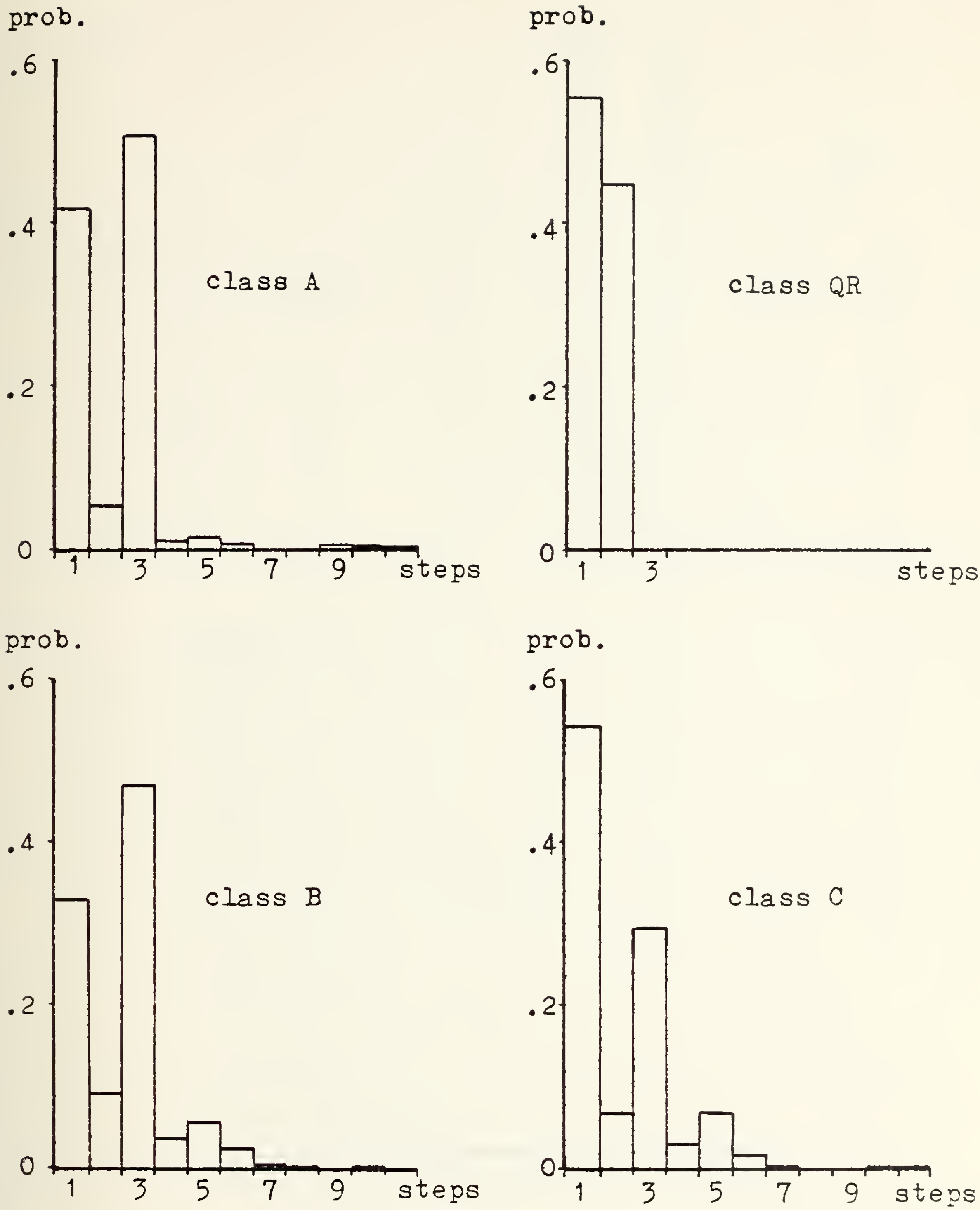


Figure 12 - HISTOGRAM: JOB STEPS PER CLASS (1)

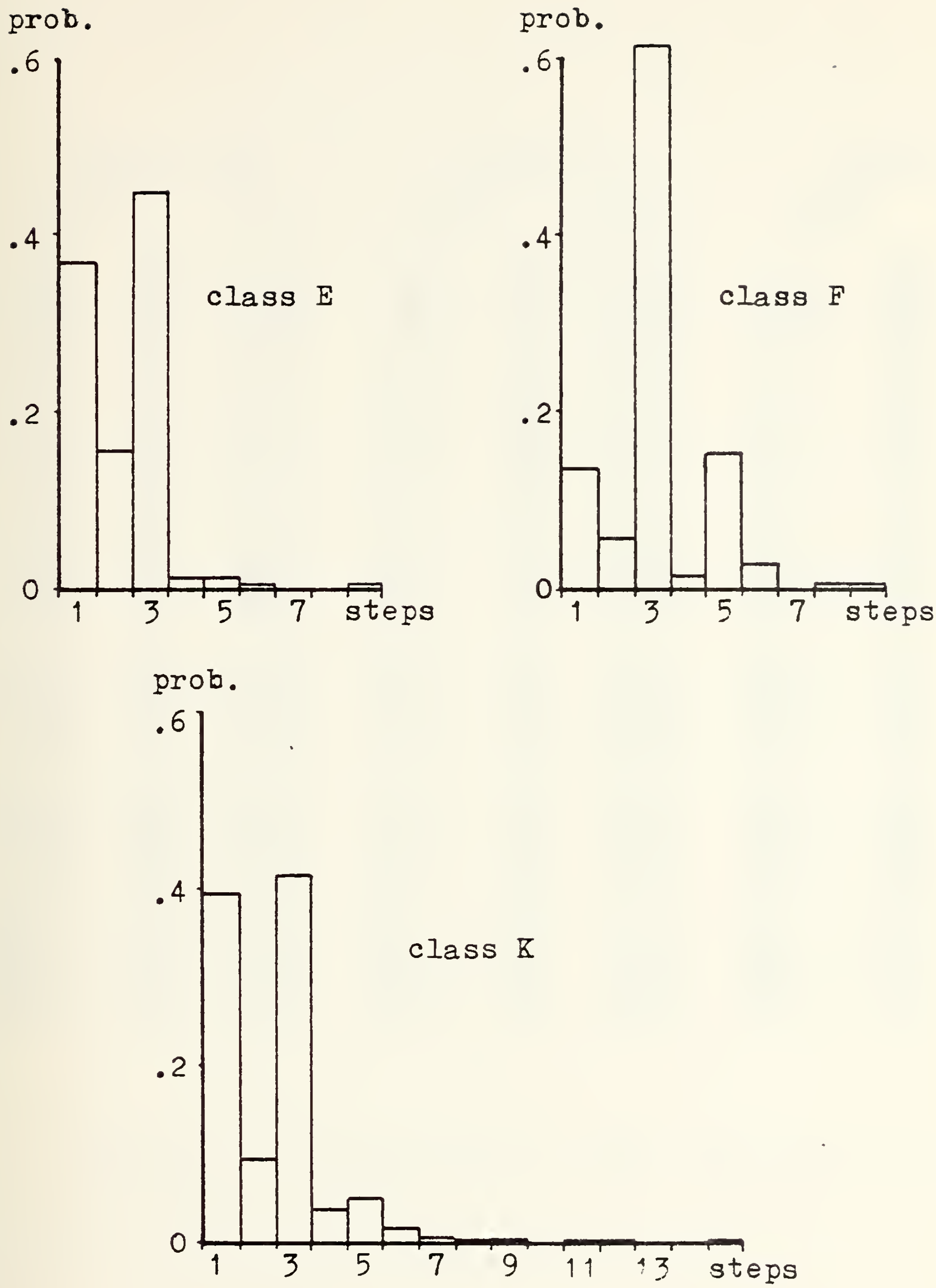


Figure 13 - HISTOGRAM: JOB STEPS PER CLASS (2)

CLASS A,QR		CLASS B		CLASS C	
CARDS	PROB.	CARDS	PROB.	CARDS	PROB.
0 - 39	.3385	0 - 39	.4004	0 - 39	.4205
40 - 79	.1686	40 - 79	.1271	40 - 79	.1646
80 - 119	.0807	80 - 119	.0727	80 - 119	.0752
120 - 159	.0687	120 - 159	.0467	120 - 159	.0659
160 - 199	.0593	160 - 199	.0486	160 - 199	.0410
200 - 239	.0617	200 - 239	.0409	200 - 239	.0232
240 - 279	.0415	240 - 279	.0417	240 - 279	.0162
280 - 319	.0307	280 - 319	.0248	280 - 319	.0381
320 - 359	.0247	320 - 359	.0180	320 - 359	.0109
360 - 399	.0244	360 - 399	.0307	360 - 399	.0113
400 - 439	.0202	400 - 439	.0135	400 - 439	.0116
440 - 479	.0107	440 - 479	.0103	440 - 479	.0251
480 - 519	.0103	480 - 519	.0128	480 - 519	.0077
520 - 559	.0063	520 - 559	.0124	520 - 559	.0086
560 - 599	.0055	560 - 599	.0086	560 - 599	.0073
600 - 639	.0029	600 - 639	.0088	600 - 639	.0069
640 - 679	.0040	640 - 679	.0055	640 - 679	.0083
680 - 719	.0043	680 - 719	.0086	680 - 719	.0036
720 - 759	.0025	720 - 759	.0097	720 - 759	.0017
760 - 799	.0051	760 - 799	.0071	760 - 799	.0023
800 - 839	.0058	800 - 839	.0057	800 - 839	.0033
840 - 879	.0023	840 - 879	.0032	840 - 879	.0043
880 - 919	.0040	880 - 919	.0035	880 - 919	.0056
920 - 959	.0043	920 - 959	.0051	920 - 959	.0030
960 - 999	.0036	960 - 999	.0040	960 - 999	.0033
1000 - 1039	.0027	1000 - 1039	.0046	1000 - 1039	.0077
1040 - 1079	.0027	1040 - 1079	.0059	1040 - 1079	.0129
1080 - 1119	.0018	1080 - 1119	.0065	1080 - 1119	.0073
1120 - 1159	.0017	1120 - 1159	.0042	1120 - 1159	.0000
1160 - 1199	.0005	1160 - 1199	.0084	1160 - 1199	.0026

Table V - DISTRIBUTION OF INPUT CARDS PER JOB (1)

CLASS E		CLASS F		CLASS K	
CARDS	PROB.	CARDS	PROB.	CARDS	PROB.
80 - 159	.2616	80 - 159	.1558	80 - 159	.0930
160 - 239	.0846	160 - 239	.0725	160 - 239	.0854
240 - 319	.0307	240 - 319	.0616	240 - 319	.0431
320 - 399	.0577	320 - 399	.0362	320 - 399	.0423
400 - 479	.0847	400 - 479	.0580	400 - 479	.0377
480 - 559	.0423	480 - 559	.0217	480 - 559	.0169
560 - 639	.0461	560 - 639	.0217	560 - 639	.0100
640 - 719	.0231	640 - 719	.0290	640 - 719	.0069
720 - 799	.0115	720 - 799	.0399	720 - 799	.0170
800 - 879	.0000	800 - 879	.1159	800 - 879	.0184
880 - 959	.0193	880 - 959	.0218	880 - 959	.0108
960 - 1039	.0000	960 - 1039	.0181	960 - 1039	.0085
1040 - 1119	.0115	1040 - 1119	.0290	1040 - 1119	.0084
1120 - 1199	.0038	1120 - 1199	.0289	1120 - 1199	.0100
1200 - 1279	.0077	1200 - 1279	.0327	1200 - 1279	.0110
1280 - 1359	.0000	1280 - 1359	.0072	1280 - 1359	.0021
1360 - 1439	.0000	1360 - 1439	.0399	1360 - 1439	.0038
1440 - 1519	.0039	1440 - 1519	.0000	1440 - 1519	.0039
1520 - 1599	.0077	1520 - 1599	.0217	1520 - 1599	.0077
1600 - 1679	.0115	1600 - 1679	.0326	1600 - 1679	.0069
1680 - 1759	.0077	1680 - 1759	.0000	1680 - 1759	.0031
1760 - 1839	.0000	1760 - 1839	.0000	1760 - 1839	.0046
1840 - 1919	.0039	1840 - 1919	.0000	1840 - 1919	.0046
1920 - 1999	.0000	1920 - 1999	.0000	1920 - 1999	.0023
2000 - 2079	.0000	2000 - 2079	.0000	2000 - 2079	.0008
2080 - 2159	.0000	2080 - 2159	.0000	2080 - 2159	.0015
2160 - 2239	.0000	2160 - 2239	.0000	2160 - 2239	.0008
2240 - 2319	.0038	2240 - 2319	.0000	2240 - 2319	.0023
2320 - 2399	.0000	2320 - 2399	.0000	2320 - 2399	.0000

Table VI - DISTRIBUTION OF INPUT CARDS PER JOB (2)

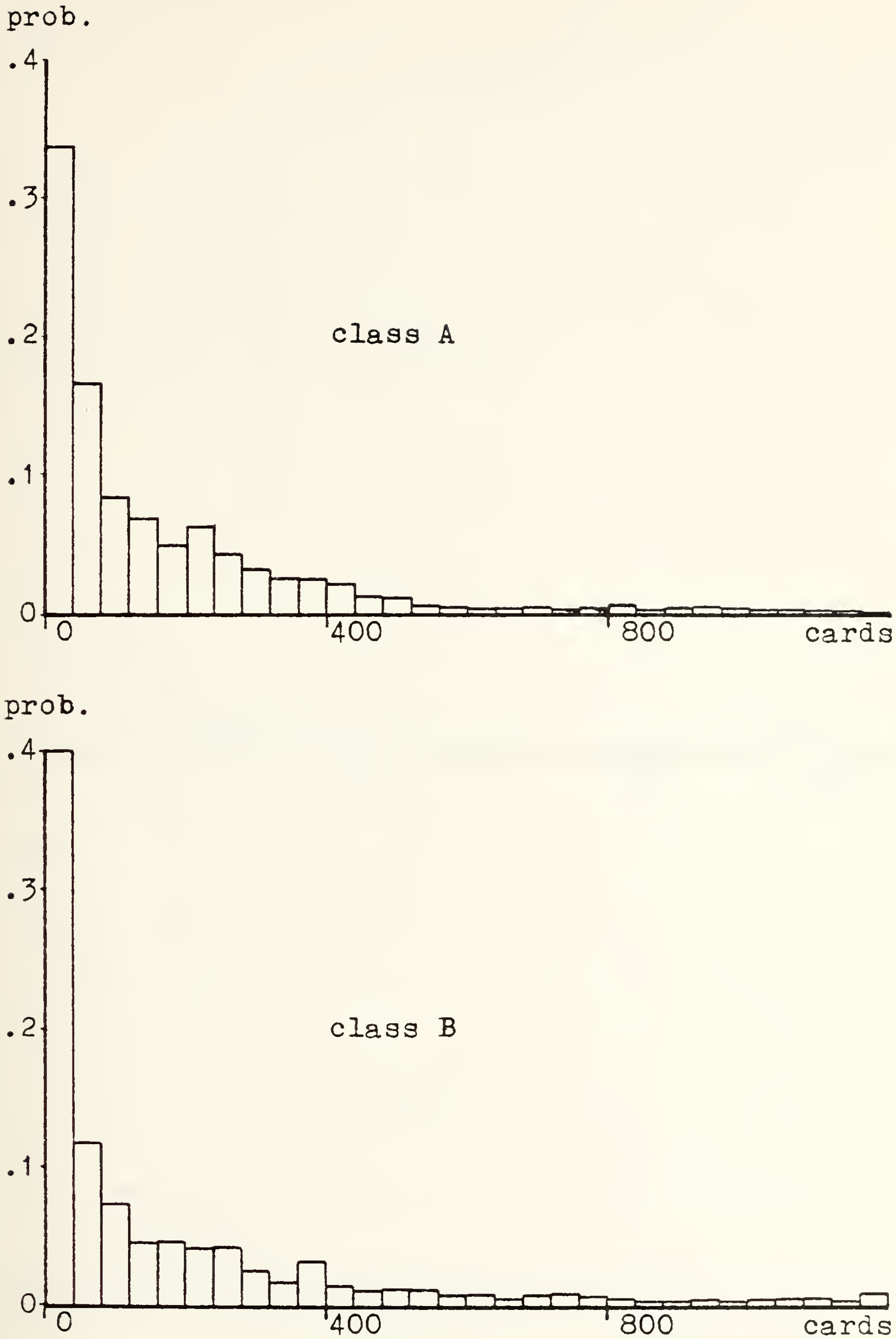


Figure 14 - HISTOGRAM: INPUT CARDS PER JOB (1)

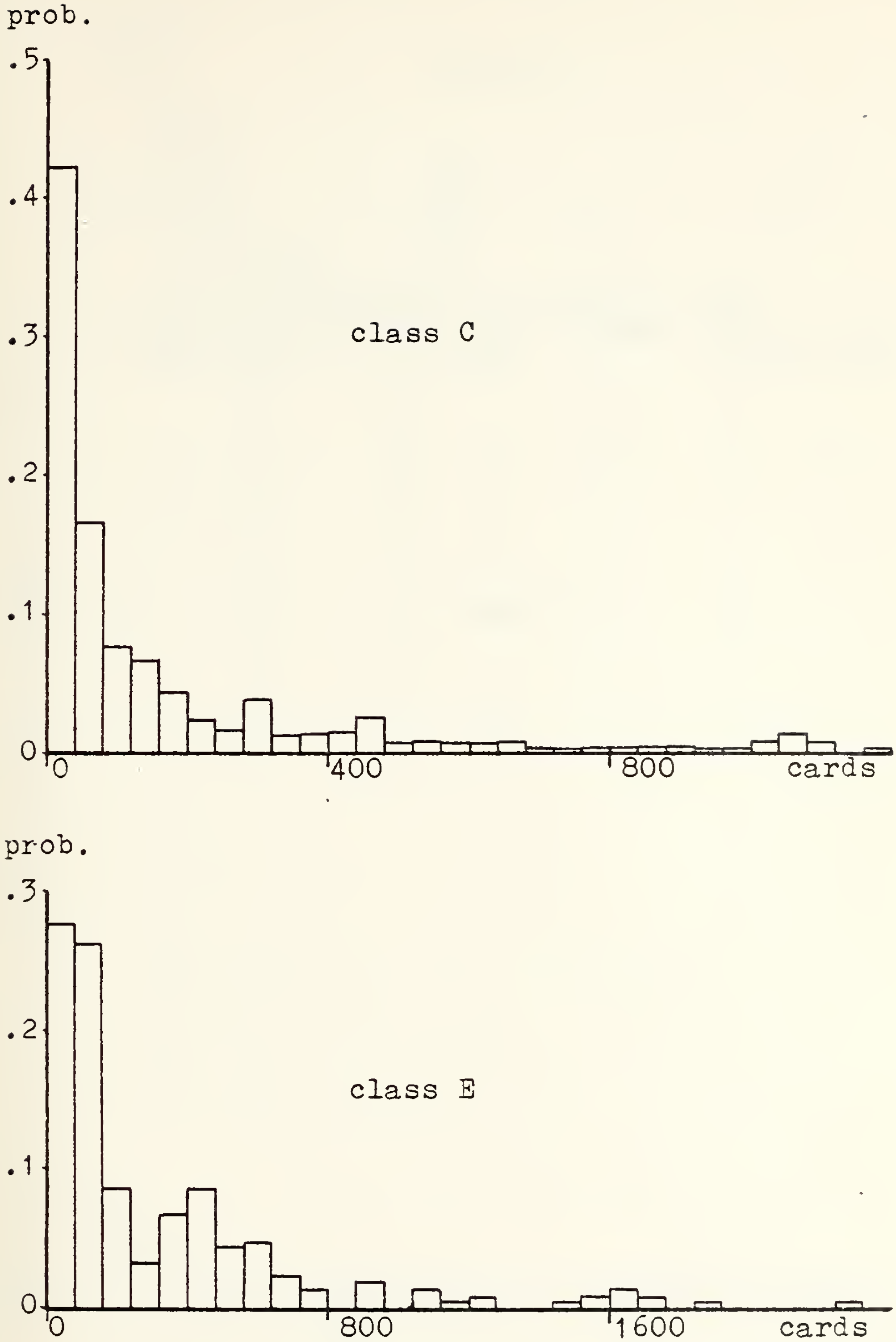


Figure 15 - HISTOGRAM: INPUT CARDS PER JOB (2)

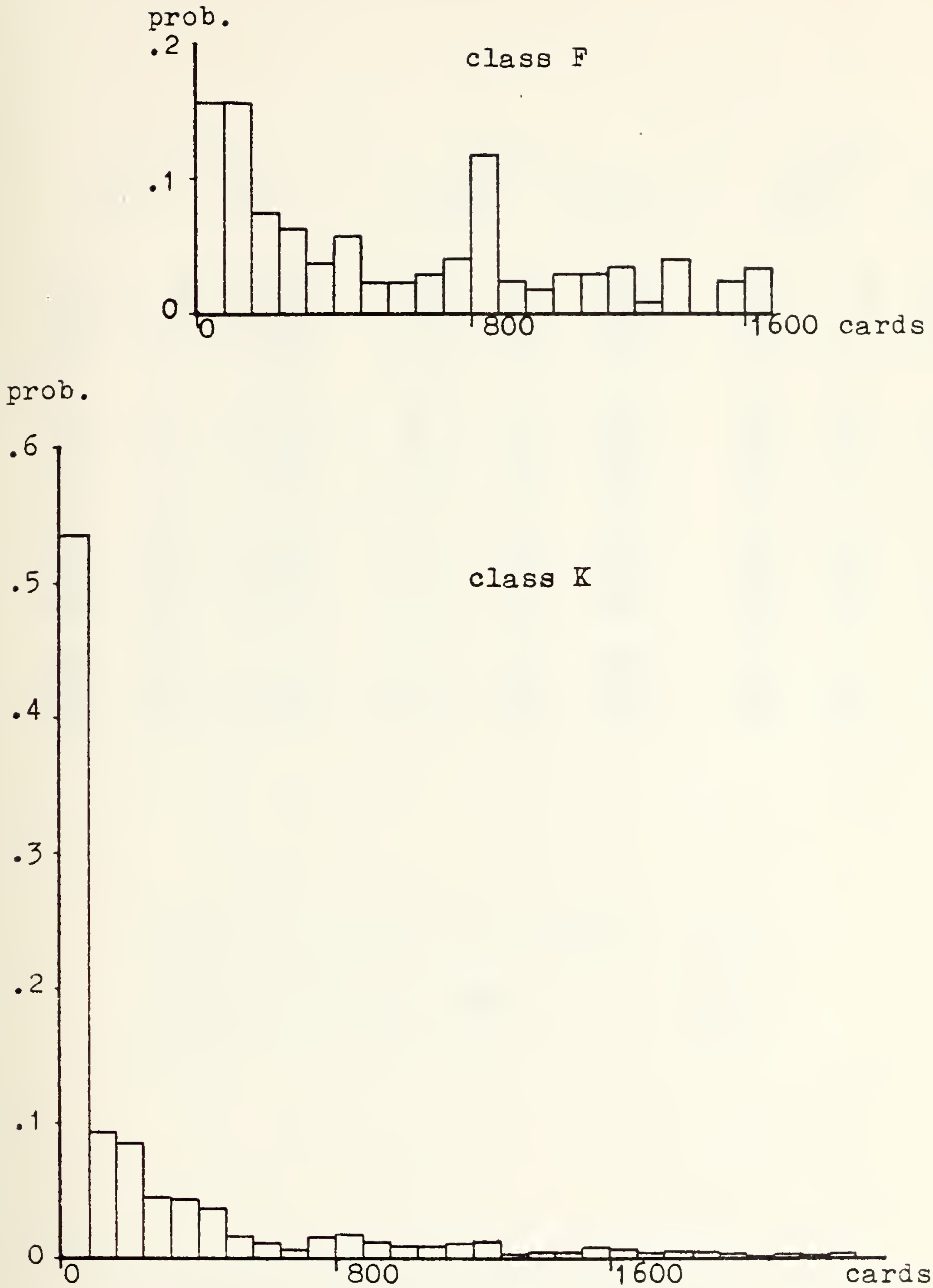


Figure 16 - HISTOGRAM: INPUT CARDS PER JOB (3)

CLASS A,QR		CLASS B		CLASS C	
CORE	PROB.	CORE	PROB.	CORE	PROB.
0 - 9	.0000	0 - 9	.0000	0 - 19	.0000
10 - 19	.0000	10 - 19	.0000	20 - 39	.0000
20 - 29	.0000	20 - 29	.0000	40 - 59	.0000
30 - 39	.0000	30 - 39	.0000	60 - 79	.1516
40 - 49	.0000	40 - 49	.0000	80 - 99	.0047
50 - 59	.0000	50 - 59	.0000	100 - 119	.3282
60 - 69	.3074	60 - 69	.2049	120 - 139	.0218
70 - 79	.0044	70 - 79	.0007	140 - 159	.0041
80 - 89	.0031	80 - 89	.0202	160 - 179	.1330
90 - 99	.0079	90 - 99	.0069	180 - 199	.0333
100 - 109	.5039	100 - 109	.4867	200 - 219	.1227
110 - 119	.0055	110 - 119	.0209	220 - 239	.0156
120 - 129	.0099	120 - 129	.0236	240 - 259	.1850
130 - 139	.0062	130 - 139	.0079	260 - 279	.0000
140 - 149	.0002	140 - 149	.0007	280 - 299	.0000
150 - 159	.1133	150 - 159	.1428	300 - 319	.0000
160 - 169	.0041	160 - 169	.0086	320 - 339	.0000
170 - 179	.0045	170 - 179	.0138	340 - 359	.0000
180 - 189	.0296	180 - 189	.0623	360 - 379	.0000

min. core size (all classes): 62 K

max. core size (class A,QR): 180 K

max. core size (class B): 180 K

max. core size (class C): 250 K

Table VII - DISTRIBUTION OF CORE USED PER STEP (1)

CLASS E CORE	PROB.	CLASS F CORE	PROB.	CLASS K CORE	PROB.
0 - 39	.0000	0 - 39	.0000	0 - 39	.0000
40 - 79	.1238	40 - 79	.1688	40 - 79	.1851
80 - 119	.3254	80 - 119	.4087	80 - 119	.3978
120 - 159	.0032	120 - 159	.0321	120 - 159	.0501
160 - 199	.0333	160 - 199	.2032	160 - 199	.1599
200 - 239	.0317	200 - 239	.0149	200 - 239	.0465
240 - 279	.0556	240 - 279	.0700	240 - 279	.0633
280 - 319	.2302	280 - 319	.0539	280 - 319	.0255
320 - 359	.1968	320 - 359	.0000	320 - 359	.0029
360 - 399	.0000	360 - 399	.0229	360 - 399	.0155
400 - 439	.0000	400 - 439	.0255	400 - 439	.0149
440 - 479	.0000	440 - 479	.0000	440 - 479	.0081
480 - 519	.0000	480 - 519	.0000	480 - 519	.0032
520 - 559	.0000	520 - 559	.0000	520 - 559	.0000
560 - 599	.0000	560 - 599	.0000	560 - 599	.0175
600 - 639	.0000	600 - 639	.0000	600 - 639	.0061
640 - 679	.0000	640 - 679	.0000	640 - 679	.0036
680 - 719	.0000	680 - 719	.0000	680 - 719	.0000
720 - 759	.0000	720 - 759	.0000	720 - 759	.0000

min. core size (all classes): 62 K

max. core size (class E): 350 K

max. core size (class F): 400 K

max. core size (class K): >400 K

Table VIII - DISTRIBUTION OF CORE USED PER STEP (2)

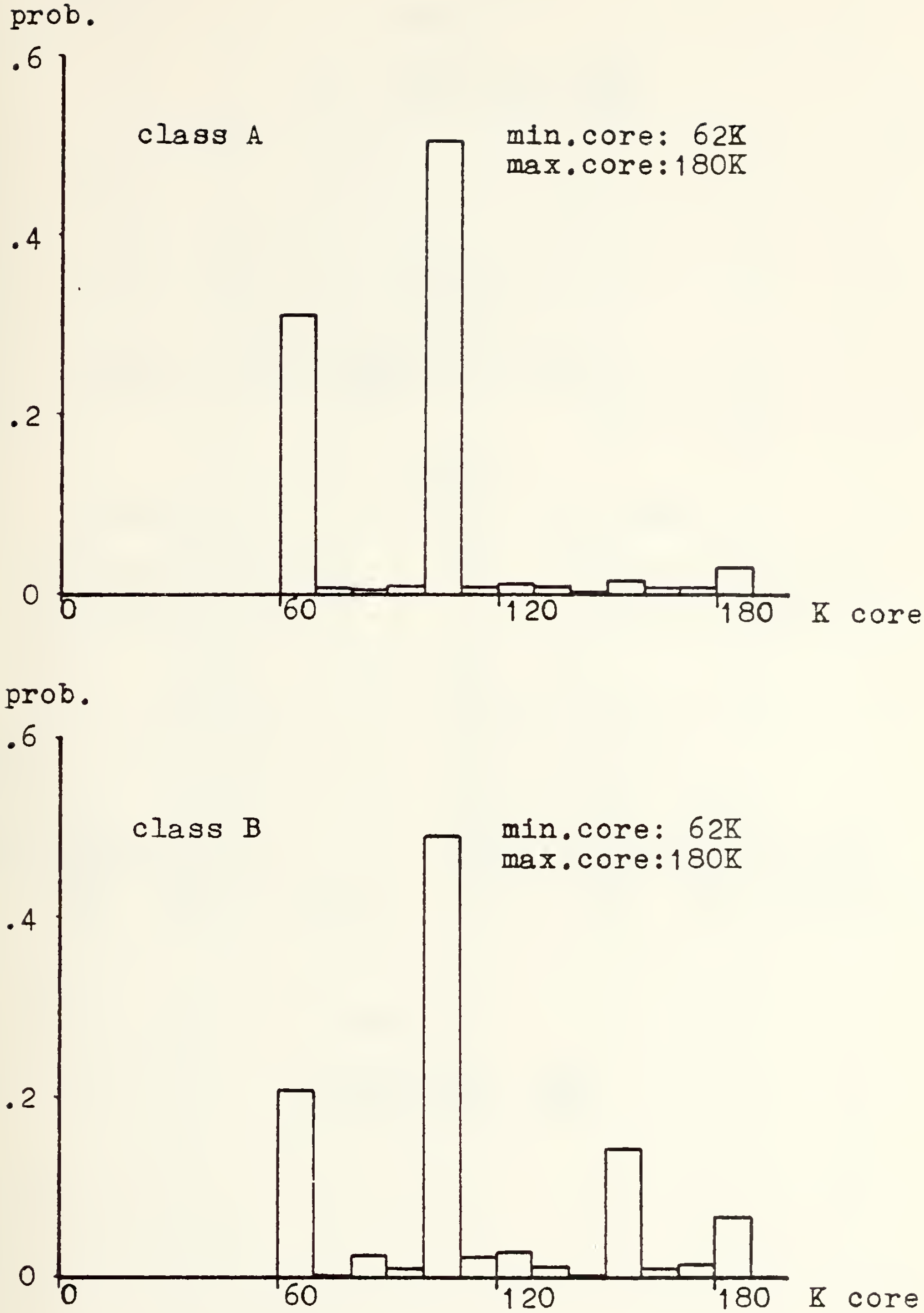


Figure 17 - HISTOGRAM: CORE USED PER STEP (1)

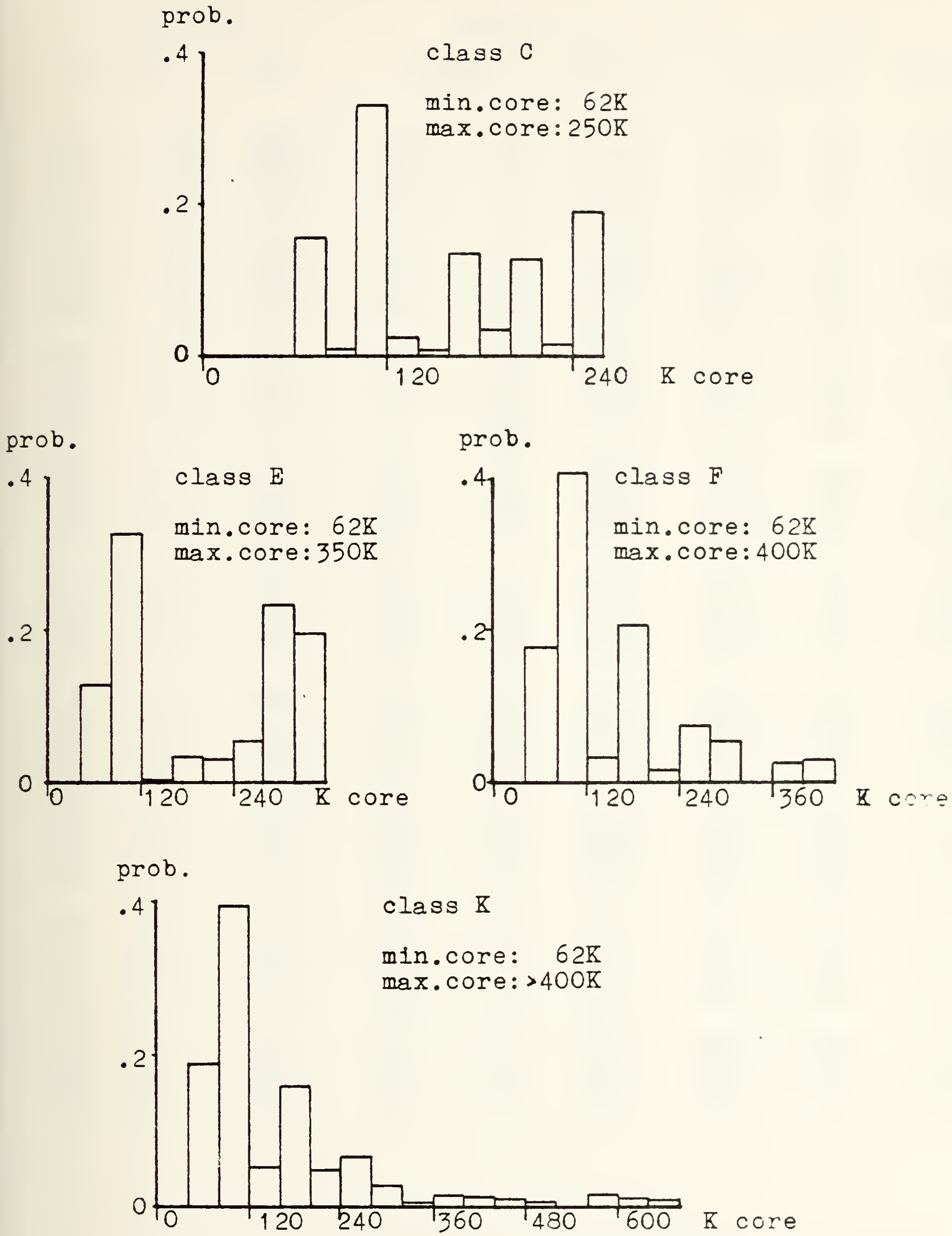


Figure 18 - HISTOGRAM: CORE USED PER STEP (2)

CLASS A,QR		CLASS B		CLASS C	
SECONDS	PROB.	SECONDS	PROB.	SECONDS	PROB.
0 - 9	.0621	0 - 19	.0962	0 - 39	.1922
10 - 19	.0630	20 - 39	.1304	40 - 79	.2134
20 - 29	.0979	40 - 59	.1359	80 - 119	.1359
30 - 39	.1128	60 - 79	.1210	120 - 159	.0925
40 - 49	.1048	80 - 99	.0963	160 - 199	.0579
50 - 59	.0943	100 - 119	.0774	200 - 239	.0454
60 - 69	.0790	120 - 139	.0549	240 - 279	.0357
70 - 79	.0595	140 - 159	.0445	280 - 319	.0289
80 - 89	.0511	160 - 179	.0371	320 - 359	.0266
90 - 99	.0437	180 - 199	.0298	360 - 399	.0194
100 - 109	.0367	200 - 219	.0243	400 - 439	.0202
110 - 119	.0307	220 - 239	.0181	440 - 479	.0145
120 - 129	.0247	240 - 259	.0164	480 - 519	.0150
130 - 139	.0210	260 - 279	.0143	520 - 559	.0122
140 - 149	.0170	280 - 299	.0124	560 - 599	.0098
150 - 159	.0132	300 - 319	.0092	600 - 639	.0078
160 - 169	.0123	320 - 339	.0090	640 - 679	.0069
170 - 179	.0096	340 - 359	.0068	680 - 719	.0067
180 - 189	.0089	360 - 379	.0058	720 - 759	.0061
190 - 199	.0064	380 - 399	.0061	760 - 799	.0057
200 - 209	.0063	400 - 419	.0053	800 - 839	.0050
210 - 219	.0051	420 - 439	.0047	840 - 879	.0047
220 - 229	.0047	440 - 459	.0050	880 - 919	.0051
230 - 239	.0036	460 - 479	.0040	920 - 959	.0058
240 - 249	.0027	480 - 499	.0032	960 - 999	.0037
250 - 259	.0032	500 - 519	.0034	1000 - 1039	.0029
260 - 269	.0028	520 - 539	.0036	1040 - 1079	.0026
270 - 279	.0023	540 - 559	.0031	1080 - 1119	.0023
280 - 289	.0021	560 - 579	.0025	1120 - 1159	.0021
290 - 299	.0018	580 - 599	.0028	1160 - 1199	.0009
300 - 309	.0019	600 - 619	.0018	1200 - 1239	.0014
310 - 319	.0016	620 - 639	.0021	1240 - 1279	.0012
320 - 329	.0016	640 - 659	.0018	1280 - 1319	.0017
330 - 339	.0015	660 - 679	.0021	1320 - 1359	.0021
340 - 349	.0013	680 - 699	.0014	1360 - 1399	.0016
350 - 359	.0017	700 - 719	.0018	1400 - 1439	.0009
360 - 369	.0012	720 - 739	.0014	1440 - 1479	.0017
370 - 379	.0009	740 - 759	.0014	1480 - 1519	.0015
380 - 389	.0013	760 - 779	.0013	1520 - 1559	.0000
390 - 399	.0012	780 - 799	.0014	1560 - 1599	.0000
400 - 409	.0005	800 - 819	.0000	1600 - 1639	.0000
410 - 419	.0010	820 - 839	.0000	1640 - 1679	.0000
420 - 429	.0006	840 - 859	.0000	1680 - 1719	.0000
430 - 439	.0004	860 - 879	.0000	1720 - 1759	.0000
440 - 449	.0000	880 - 899	.0000	1760 - 1799	.0000

Table IX - DISTRIBUTION OF ELAPSED STEP RUN TIME (1)

CLASS E SECONDS	PROB.	CLASS F SECONDS	PROB.	CLASS K SECONDS	PROB.
0 - 39	.2107	0 - 39	.3698	0 - 79	.4677
40 - 79	.2593	40 - 79	.2117	80 - 159	.1680
80 - 119	.1313	80 - 119	.0998	160 - 239	.0710
120 - 159	.0778	120 - 159	.0425	240 - 319	.0512
160 - 199	.0600	160 - 199	.0402	320 - 399	.0357
200 - 239	.0421	200 - 239	.0280	400 - 479	.0229
240 - 279	.0276	240 - 279	.0170	480 - 559	.0151
280 - 319	.0275	280 - 319	.0195	560 - 639	.0341
320 - 359	.0178	320 - 359	.0097	640 - 719	.0151
360 - 399	.0195	360 - 399	.0158	720 - 799	.0138
400 - 439	.0129	400 - 439	.0122	800 - 879	.0121
440 - 479	.0130	440 - 479	.0146	880 - 959	.0078
480 - 519	.0130	480 - 519	.0146	960 - 1039	.0087
520 - 559	.0129	520 - 559	.0146	1040 - 1119	.0091
560 - 599	.0081	560 - 599	.0085	1120 - 1199	.0084
600 - 639	.0065	600 - 639	.0122	1200 - 1279	.0058
640 - 679	.0081	640 - 679	.0121	1280 - 1359	.0064
680 - 719	.0049	680 - 719	.0110	1360 - 1439	.0053
720 - 759	.0081	720 - 759	.0061	1440 - 1519	.0041
760 - 799	.0032	760 - 799	.0036	1520 - 1599	.0054
800 - 839	.0081	800 - 839	.0061	1600 - 1679	.0033
840 - 879	.0033	840 - 879	.0036	1680 - 1759	.0031
880 - 919	.0064	880 - 919	.0061	1760 - 1839	.0040
920 - 959	.0049	920 - 959	.0049	1840 - 1919	.0047
960 - 999	.0068	960 - 999	.0049	1920 - 1999	.0031
1000 - 1039	.0149	1000 - 1039	.0012	2000 - 2079	.0023
1040 - 1079	.0032	1040 - 1079	.0000	2080 - 2159	.0027
1080 - 1119	.0017	1080 - 1119	.0036	2160 - 2239	.0037
1120 - 1159	.0000	1120 - 1159	.0037	2240 - 2319	.0020
1160 - 1199	.0000	1160 - 1199	.0024	2320 - 2399	.0034

Table X - DISTRIBUTION OF ELAPSED STEP RUN TIME (2)

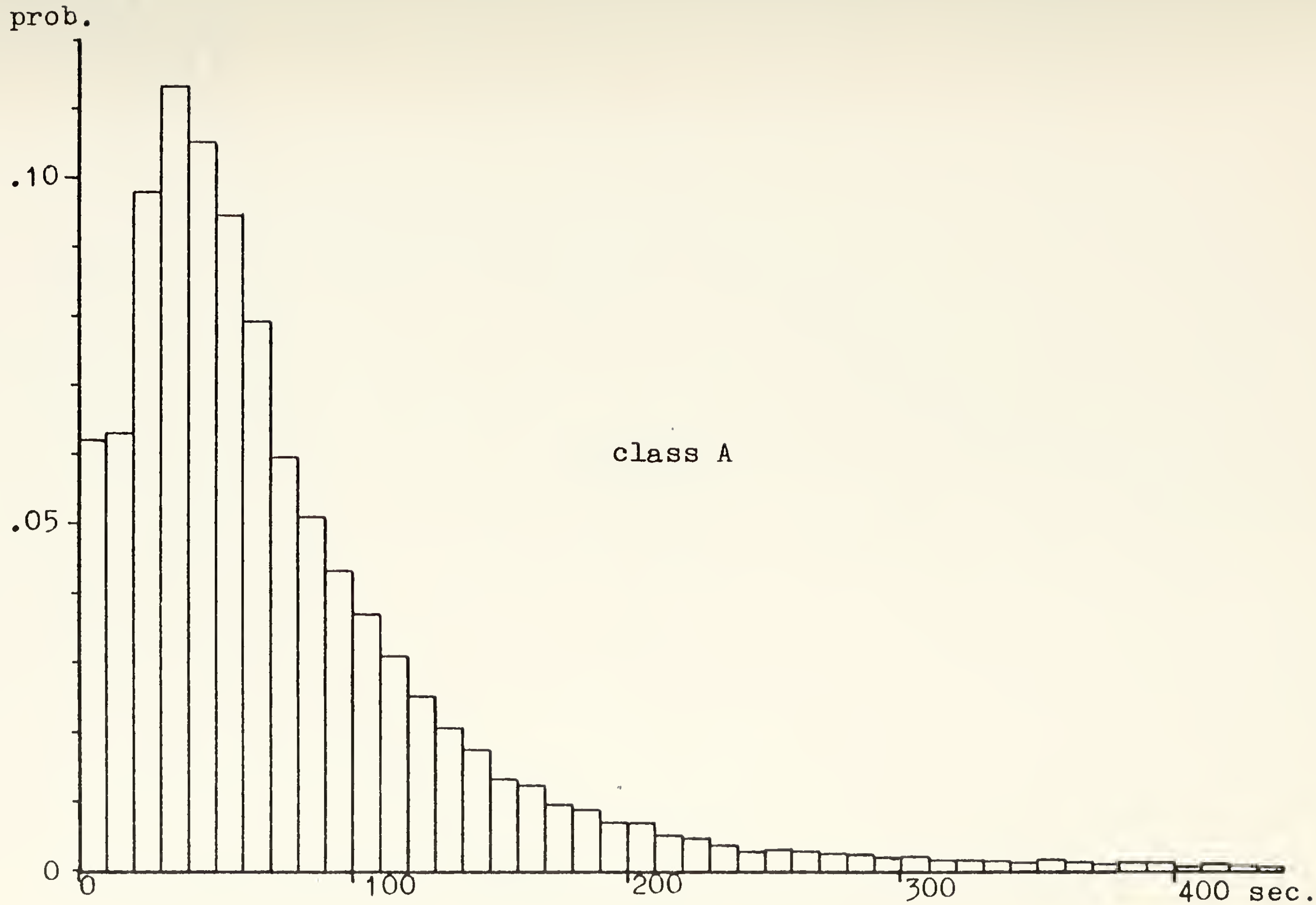


Figure 19 - HISTOGRAM: ELAPSED STEP RUN TIME (1)

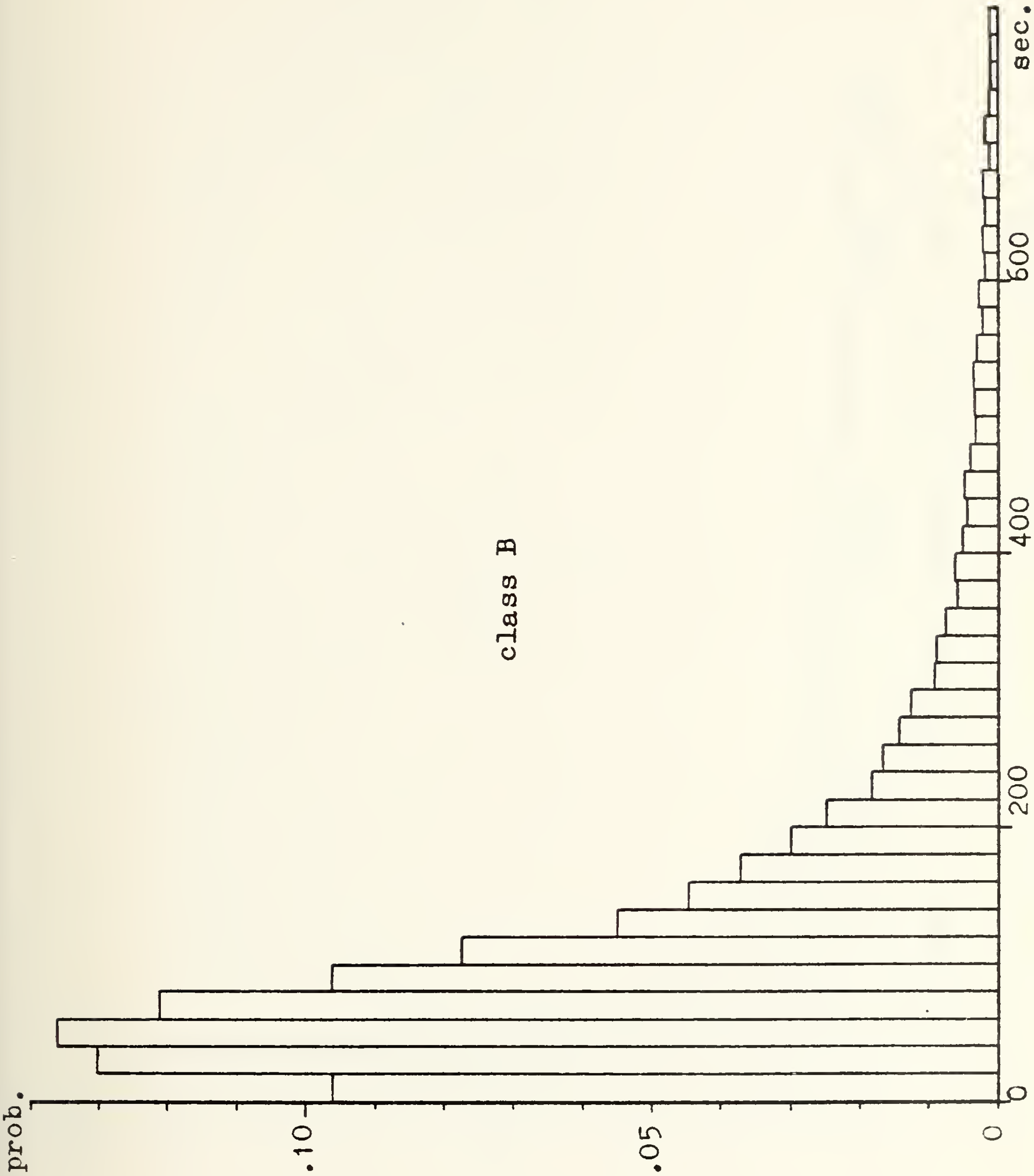


Figure 20 - HISTOGRAM: ELAPSED STEP RUN TIME (2)

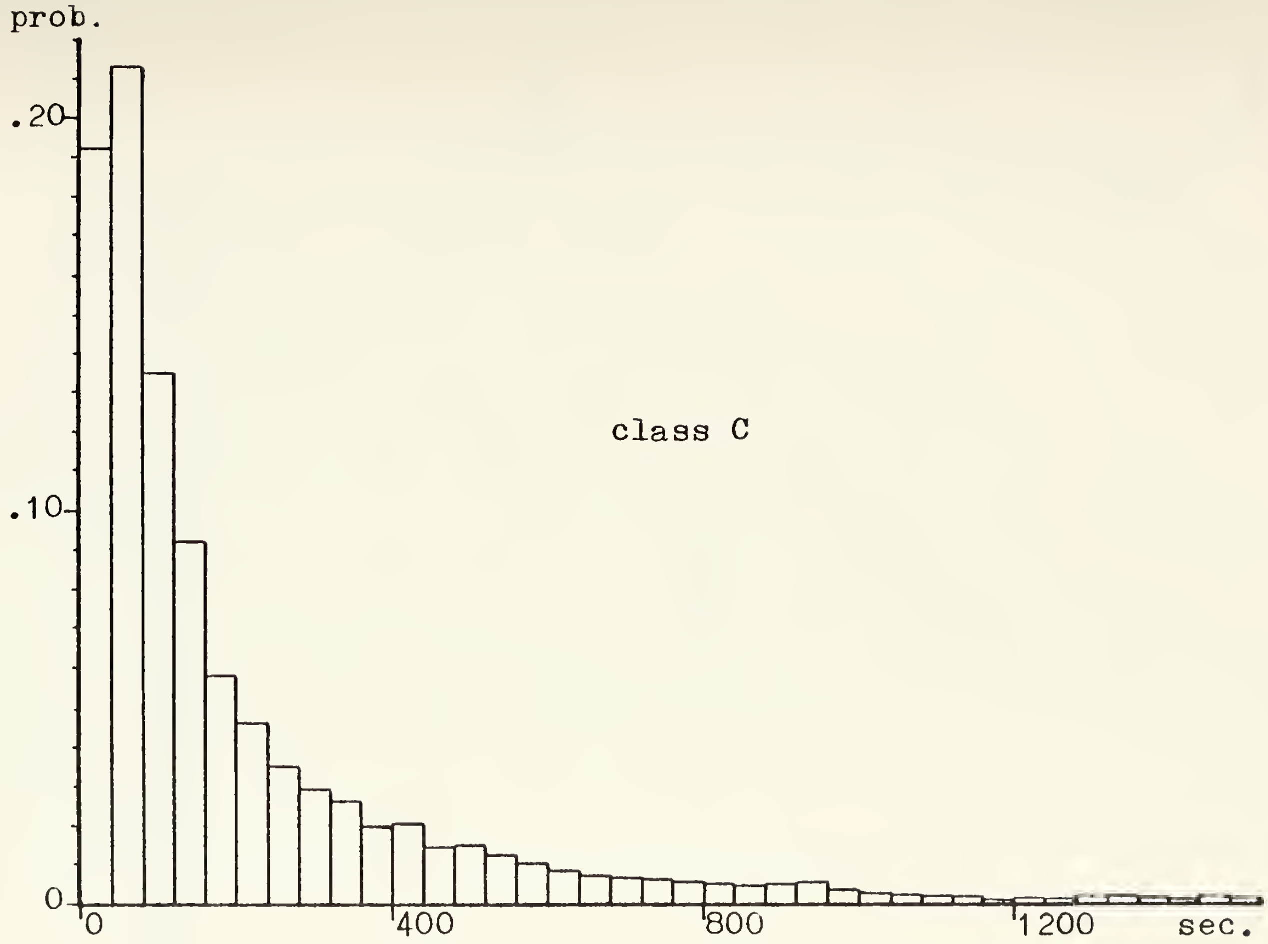


Figure 21 - HISTOGRAM: ELAPSED STEP RUN TIME (3)

Figure 22 - HISTOGRAM: ELAPSED STEP RUN TIME (4)

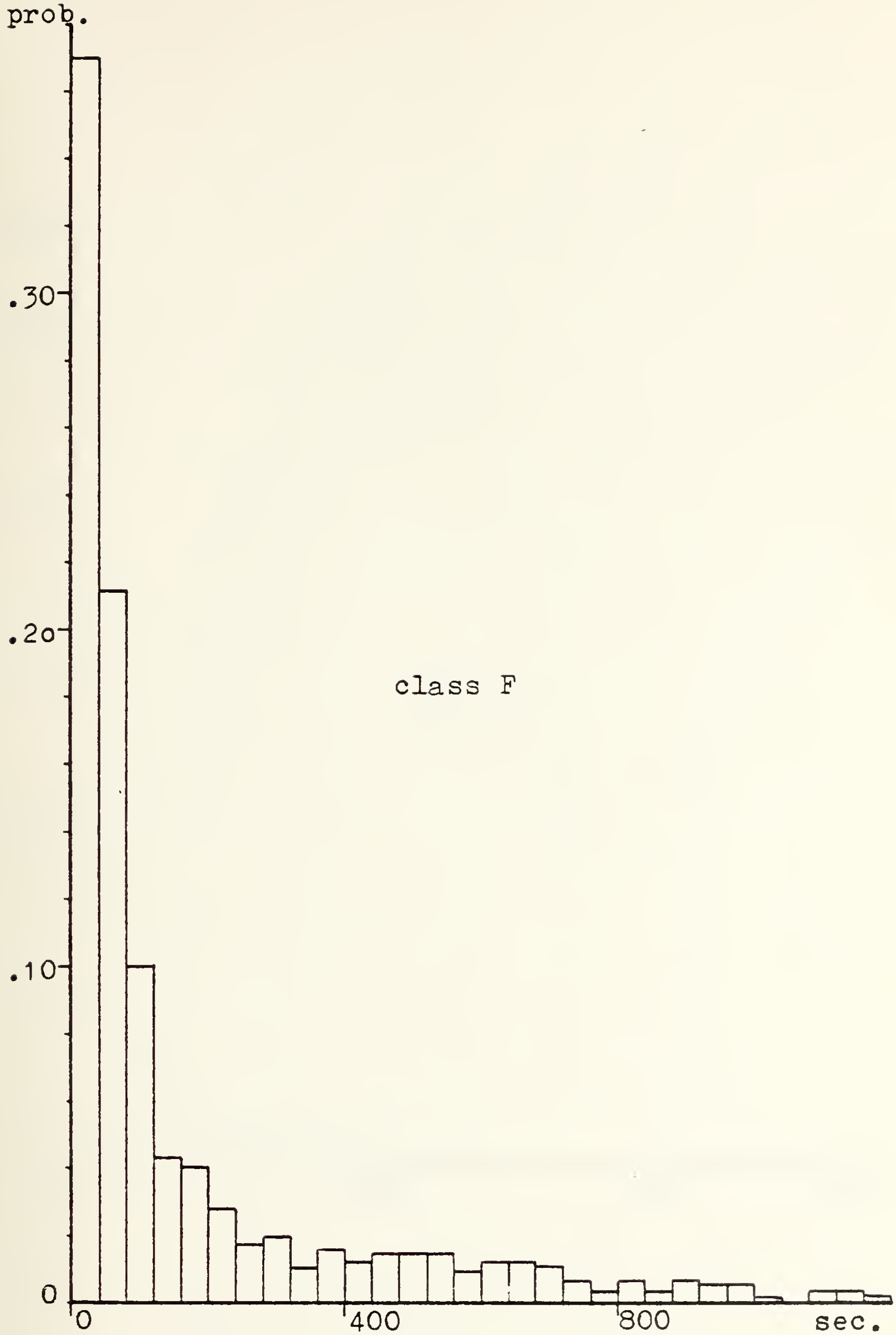


Figure 23 - HISTOGRAM: ELAPSED STEP RUN TIME (5)

Figure 24 - HISTOGRAM: ELAPSED STEP RUN TIME (6)

		CLASS			
TAPES	DISKS	B	C	E	K
0	0	.8887	.8887	.8887	.8570
0	1	.0395	.0395	.0395	.0381
0	2	.0019	.0019	.0019	.0018
1	1	.0019	.0019	.0019	.0018
2	1	.0005	.0005	.0005	.0005
2	2	.0003	.0003	.0003	.0003
1	3	.0003	.0003	.0003	.0003
2	3	.0005	.0005	.0005	.0005
1	0	.0584	.0584	.0584	.0563
2	0	.0079	.0079	.0079	.0076
3	0	.0000	.0000	.0000	.0357

Table XI - DISTRIBUTION OF TAPES AND DISKS PER JOB CLASS

C. OPERATOR RESPONSE TIMES

The system logs were also used to evaluate the operator volume mounting times, their response times to other system requests, and the number of jobs cancelled by the operators because a request could not be satisfied. One problem for the evaluation was the fact that the system requests had no time stamps. In most cases the time could be estimated within a 10 second range from other system messages with time stamps just above and below the request messages. For tape and disk mounts there were also no direct operator answers on the system logs, but in a certain number of cases the actual mounting time could be estimated from other system messages. Here again only those cases were evaluated where the estimation could be made within a 10 second time range. Using this aproach a total of about 700 operator response times could be used. The probability distribution per job step, separated into the cases for tape mount, disk mount, and other system requests, is given in Table XII; a histogram is provided in Figure 25.

The relativly high probability of short reaction times to tape and disk mount requests came from the fact that the requested volumes were already pre-mounted and the devices had only to be varied on-line.

The number of jobs cancelled by the operators because a request could not be satisfied could be counted exactly: 49 jobs or 1.31 % out of 3,735 jobs.

TAPE MOUNT REQUESTS		DISK MOUNT REQUESTS		OTHER REQUESTS	
SECONDS	PROB.	SECONDS	PROB.	SECONDS	PROB.
0 - 19	.0000	0 - 19	.0000	0 - 19	.6715
20 - 39	.0733	20 - 39	.0197	20 - 39	.0253
40 - 59	.1721	40 - 59	.0527	40 - 59	.0433
60 - 79	.1795	60 - 79	.0789	60 - 79	.0577
80 - 99	.1172	80 - 99	.1316	80 - 99	.0181
100 - 119	.0843	100 - 119	.0658	100 - 119	.0108
120 - 139	.0732	120 - 139	.1381	120 - 139	.0217
140 - 159	.0257	140 - 159	.0856	140 - 159	.0142
160 - 179	.0256	160 - 179	.0526	160 - 179	.0074
180 - 199	.0330	180 - 199	.0592	180 - 199	.0217
200 - 219	.0329	200 - 219	.0592	200 - 219	.0036
220 - 239	.0184	220 - 239	.0132	220 - 239	.0037
240 - 259	.0329	240 - 259	.0395	240 - 259	.0036
260 - 279	.0147	260 - 279	.0131	260 - 279	.0036
280 - 299	.0110	280 - 299	.0263	280 - 299	.0036
300 - 319	.0183	300 - 319	.0132	300 - 319	.0036
320 - 339	.0146	320 - 339	.0131	320 - 339	.0036
340 - 359	.0110	340 - 359	.0198	340 - 359	.0036
360 - 379	.0074	360 - 379	.0263	360 - 379	.0036
380 - 399	.0073	380 - 399	.0197	380 - 399	.0036
400 - 419	.0027	400 - 419	.0066	400 - 419	.0036
420 - 439	.0173	420 - 439	.0061	420 - 439	.0036
440 - 459	.0037	440 - 459	.0071	440 - 459	.0037
460 - 479	.0037	460 - 479	.0065	460 - 479	.0036
480 - 499	.0036	480 - 499	.0066	480 - 499	.0036
500 - 519	.0037	500 - 519	.0066	500 - 519	.0036
520 - 539	.0036	520 - 539	.0066	520 - 539	.0036
540 - 559	.0037	540 - 559	.0066	540 - 559	.0036
560 - 579	.0037	560 - 579	.0065	560 - 579	.0036
580 - 599	.0036	580 - 599	.0066	580 - 599	.0036
600 - 619	.0037	600 - 619	.0066	600 - 619	.0036
620 - 639	.0000	620 - 639	.0000	620 - 639	.0036
640 - 659	.0000	640 - 659	.0000	640 - 659	.0036
660 - 679	.0000	660 - 679	.0000	660 - 679	.0037
680 - 699	.0000	680 - 699	.0000	680 - 699	.0036
700 - 719	.0000	700 - 719	.0000	700 - 719	.0036
720 - 739	.0000	720 - 739	.0000	720 - 739	.0036
740 - 759	.0000	740 - 759	.0000	740 - 759	.0036
760 - 779	.0000	760 - 779	.0000	760 - 779	.0036
780 - 799	.0000	780 - 799	.0000	780 - 799	.0036

Table XII - DISTRIBUTION OF OPERATOR RESPONSE TIMES

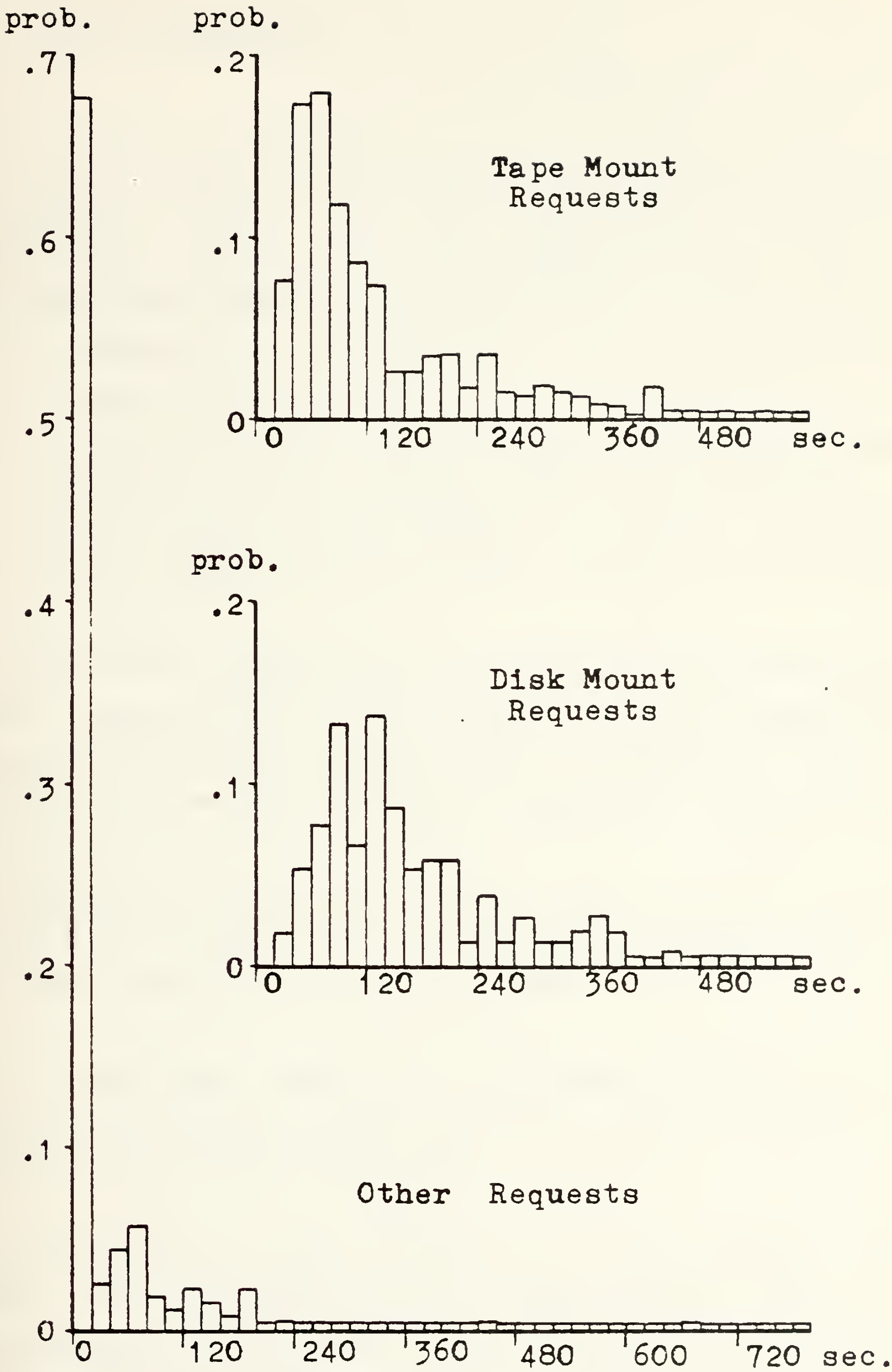


Figure 25 - HISTOGRAM: OPERATOR RESPONSE TIMES

D. SYSTEM PARAMETERS

The following system parameters were used to tailor the simulation model to the environment at the Naval Postgraduate school as it was available to the user during the time period April to August 1976:

* number of tape drives: 9
* number of disk drives: 3
* amount of input spool space: 45,000 card images
* amount of direct access space: 100 records
* main memory (high address) : 1140 K
* main memory (low address) : 140 K

The actual number of disk drives in the system was much higher (see Table I), but without special arrangements only three were free for general users. Also only five of the eight core boxes were routinely available for OS/MVT.

Assuming a mean of 300 input cards per jobs the amount of 45,000 card images was equivalent to the current system spooling capacity of about 150 jobs.

As mentioned earlier the direct access space was not used as parameter for the simulation runs. Thus the number of 100 records had no meaning.

The upper and lower address of main memory were the bounds of the Dynamic Area. These bounds varied depending on the load on the system. The values of the bounds used were mean values observed from the system logs.

V. VALIDATION

In order to show the usefullness and validity of the simulation model it was parameterized to match the characteristics of the computer center installation at the Naval Postgraduate School. The parameters used for input job stream characteristics, system configuration, and operator response times were mostly the same as described in the previous chapter. The outcome of the simulation runs could be compared with data observed from the actual system.

An unexpected problem arose when searching for console log data which could be compared with simulation results. Within August 1976, the only month for which both SMF tapes and system logs were available, there were 15 days which qualified for use in the model (more than 500 job arrivals in the period from 10 a.m. to 5 p.m.). At first this seemed to be a sufficient number of days to choose from, but a more detailed examination showed that none of these days could be used. For each day there was either system down time, or the operators held the queues up to 50 minutes, or both. In addition, the operators reset up to 40 jobs daily from one class into another or changed job priorities. The longest continuous time interval without down time, or queue hold, or with few resets was 4.5 hours. It was observed from 10:00 a.m. to 2:30 p.m. on August 16 1976. This was a rather short time period for validation purposes, but for lack of better data it had to be used.

The job arrival rate (1.2407 jobs per min.) and the job class distribution (see Table XIII) within this time interval differed significantly from the values observed over the three month period. The appropriate modification in the simulation model was made.

CLASS:	A	B	C	E	F	K	QR
PROB.:	.3403	.1940	.1045	.0179	.0149	.0716	.2567

Table XIII - DISTRIBUTION OF JOB CLASSES (VALIDATION RUNS)

Table XIV shows the usage of Initiators and their associated job classes during the validation runs. This set-up differed only in two minor points from the actual usage: Class O in the validation runs represented the old Quickrun class and class K was used in the validation runs instead of class M.

	TIME			
Initiator	10:00	12:00	12:18	14:30
1	oab	oab	oab	oab
2	oab	oab	oab	oab
3	oabc	oabc	oabc	oabc
4	oabc	oabc	oabc	oabc
5	oabce	oabce	oabce	oabce
6	k	kabfec	kabfec	kabfec
7	-	-	ab	ab

Table XIV - INITIATOR USAGE (VALIDATION RUNS)

Forty validation runs with different input job streams were made. A comparison between the actual values and the mean values from the simulations is given in Table XV. More jobs were started in some classes than arrived because the queues were partly filled with jobs which had arrived during the previous hour.

Actual Data:

	A	B	C	E	F	K	QR	total
				CLASS				
A:	114	65	35	6	5	24	86	335
S:	126	63	32	0	3	9	86	319
R:	1.1053	.9692	.9143	.0000	.6000	.3750	1.0000	.9522

Validation Results:

	A	B	C	E	F	K	QR	total
				CLASS				
A:	127	69	40	5	7	29	77	354
S:	131	71	39	3	1	20	7	343
R:	1.0270	1.0195	.9681	.6212	.1579	.7114	1.0083	.9689

A: Number of jobs arrived
S: Number of jobs started
R: Ratio jobs started to jobs arrived

Table XV - VALIDATION RESULTS

The ratio of jobs started to jobs arrived observed from the evaluation runs was very close to the actual ratio for the job class O (=Quickrun) and for the total. Good results were also obtained for classes A, B, and C. Since the sample size for classes E and F was small the results were meaningless. Class K results were not representative since in the actual system class M was used for K class jobs and these jobs were selected by the operator.

Due to lack of more usable data no futher comparison against actual system performance could be made. The small sample size available for this kind of validation did not allow a definitive statement about the accuracy of the results.

Numerous additional validation runs have been made to check individual components of the model (region management, device allocation, etc.) and to test boundary conditions (limitation in number of devices, core size, etc.) . All of these runs showed the expected results.

However, one unusual result was observed. Although the job arrival distribution generated by the simulation model closely approximated the desired distribution for a sample size of 10,000 jobs, the job arrival rate for the first 600 jobs was always too high for a given seed. To overcome this anomaly a new feature was added to the model. Upon user's request the seed for the random number generator was modified by the value of the computer clock. It was then possible to change the seed at random. When this feature was used in additional simulation runs the unusual statistical pattern was no longer observed.

Since the future use of the simulation model is to compare the relative merits of different Initiator strategies rather than to predict absolute performance, it was sufficient to assure that the principal characteristics of the Job Management functions were reasonably well simulated. The results so far demonstrate the correct functioning of the simulation model.

APPENDIX A

USER'S MANUAL

Described in this manual is the use of the simulation model under the Control Program / Cambridge Monitor System (CP/CMS) at the Naval Postgraduate School. The user should have some private CP/CMS space (P-disk) and should be familiar with the basic functions and commands of this time-sharing system.

The examples of CP/CMS commands are from an actual run. They show how to prepare and run the model and how to get results.

TABLE OF CONTENTS

I. PREPARING THE SIMULATION MODEL

A. GENERAL REMARKS

Before working with the simulation model a PROFILE EXEC file should be prepared on P-disk by the user. In the following example the PROFILE used during the simulation is printed.

```
print profile exec
&TYPEOUT OFF
GLOBAL TXTLIB PLILIB
VSET RDYMSG OFF
BLIP *
R;
```

The GLOBAL command is necessary to establish the linkage to certain libraries used at compile and run time. The other commands are optional. The TYPEOUT OFF command suppresses the typing of the PROFILE commands. The command VSET RDYMSG OFF is used to abreviate the system's error and ready messages. The BLIP command prints an asterisk every two seconds of CPU time used. With means of that the user can see the duration of the different program parts.

Once the PROFILE EXEC file is prepared on the P-disk it is executed automatically by the system whenever the user logs into CP/CMS and executes the first CMS command.

B. LOGIN PROCEDURE

The simulation model needs about 500 K main memory for the program itself and for I/O buffers. This amount of virtual storage must be requested at log-in time.

```
login 0860p10 500k
ENTER PASSWORD:

********
ENTER 4-DIGIT PROJECT NUMBER FOLLOWED BY ...
0748cr62
FILES:- 02 RDR, NO PRT, NO PUN
READY AT 10.12.16 ON 05/09/77
CMS VERSION 3.2
```

C. REQUESTING TEMPORARY DISK SPACE

With the following sequence of commands the user will get 7 additional cylinders of temporary space on a B-disk.

```
cp define t2314 192 7
R;

login 192 b
IOERR R 0794 NRF-MADDMK      ADDRESS: 192
** B (192) DEVICE ERROR **
E(00001)

format b all
** "FORMAT B" WILL ERASE ALL YOUR B-DISK (192) FILES **
** DO YOU WISH TO CONTINUE?  ENTER "YES" OR "NO":
yes
ENTER 6-BYTE LABEL (IF WANTED), OR NULL LINE (IF NOT):

FORMATTING B-DISK (2314)...
B (192): 007 CYL
R;

release 192 b
R;

login 192 p
192  REPLACES P (191)
R;

login 191 b,p
B (191) R/O
R;
```

D. READING THE PROGRAMS

It is assumed that the simulation program (SIM) and the supporting statistical evaluation program (STA) are available as card decks and are prepared for an OFFLINE READ.

```
o read sim pli
R;

o read sta pli
R;
```

E. COMPILING THE PROGRAMS

With the following commands the programs will be compiled. The option FE will suppress compiler warnings.

```
pli sim ( fe )
*********************************************************
E(00004)

pli sta ( fe )
**************E(00004)

list
FILENAME FILETYPE MODE NO.REC. DATE
SIM      PLI      P1     177   5/09
STA      PLI      P1      40   5/09
SIM      LISTING  P1     360   5/09
SIM      TEXT     P1     177   5/09
STA      LISTING  P1      88   5/09
STA      TEXT     P1      44   5/09
R;

stat
P (192): 6 FILES; 901 REC IN USE, 147 LEFT (OF 1048),
R;                                 86% FULL (7 CYL)
```

With an OFFLINE PRINT command the user can get now a compiler listing. If a listing is not wanted at all the NOPRINT (NP) option should be used to speed up the compilation. [Example : pli sim (fe np)].

Depending upon the current number of users logged into CP/CMS and upon the used compiler options the compilation of SIM will take between 7 and 30 minutes, the compilation of STA between 3 and 15 minutes wall clock time.

F. SAVING THE TEXT FILES

With the following commands the TEXT files (i.e. the compiled version of the programs) are transferred to the P-disk and the temporary disk space is released:

```
cp xfer d to 0860p
R;

o punchcc sim text
** CARDS XFERED BY 0860P10 **
** CARDS XFERED TO 0860P **
R;

o punchcc sta text
** CARDS XFERED BY 0860P10 **
** CARDS XFERED TO 0860P **
R;

cp xfer d off
R;

release 191 b
R;

login 191 p
191 REPLACES P (192)
R;

o read sim text
R;

o read sta text
R;

list
FILENAME FILETYPE MODE NO.REC. DATE
PROFILE  EXEC      P1       1   5/09
SIM      TEXT      P1     177   5/09
STA      TEXT      P1      44   5/09
R;

stat
P (191): 3 FILES; 230 REC IN USE, 66 LEFT (OF 296),
R;                                    78% FULL (2 CYL)
```

II. RUNNING THE SIMULATION MODEL

A. STARTING THE SIMULATION

When the user has logged into CP/CMS with 500 K and a SIM TEXT file is on his P-disk he can start the simulation program for an interactive run. To be sure to have enough disk space the user should again use temporary space. If the B-disk is already formatted from the previous compilation then exactly the following commands can be used. Otherwise the user must request and format the B-disk as described earlier.

```
login 192 p
192 REPLACES P (191)
R;

login 191 b,p
B (191) R/O
R;

erase * listing
R;

filedef sysin con blksize 80
R;

filedef sysprint con
R;

$ sim
*EXECUTION BEGINS...
```

With this sequence of commands program messages are directed to the user's console and he can enter the simulation parameters from there. However, the statistical output is written to separate files on the disk and must be processed later.

B. GENERAL RULE FOR PARAMETER ENTRIES

When entering parameters to the model one important rule applies: EVERY PARAMETER ENTRY MUST BE FOLLOWED BY AT LEAST ONE BLANK CHARACTER.

The blank character separates different entries from each other. The program requires that even after a single parameter or after the last in a sequence of parameters at least one blank character must be entered before hitting 'carriage return'. Otherwise the program will wait until the reqired blank is entered.

C. ENTERING SYSTEM MODIFICATIONS

The user is asked if he wants to modify the set-up of system parameters in the simulation model.

If his answer is 0 (or any other number exept 1) no modification is wanted and the program goes on.

If his answer is 1 then the following parameters can be entered:

* input spool capacity (in number of cards)
* public direct access space (in number of records)
* core - high address (in K)
* core - low address (in K)
* number of disk drives
* number of tape drives

The correct input format is:

```
mod.spool=nn
mod.space=nn
mod.core_h=nn
mod.core_l=nn
mod.disks=nn
mod.tapes=nn
```

In this format nn represents the appropriate number the user wants to enter. This number is not checked for validity. Entering invalid numbers (negative numbers, or core_l > core_h) will cause unpredictable program behavior and erroneous results.

One or more of the modification parameters can be entered in any order. Duplicate entries are allowed, then the last entry counts.

```
SYSTEM MODIFICATIONS? (1=YES, 0=NO)
1

ENTER:
mod.tapes=1
mod.disks=0
mod.core_h=640
mod.core_l=140
```

To end the modification mode the user must enter a NULL line (carriage return only).

D. ENTERING RUN PARAMETERS

The user must enter the number of jobs to be generated and the time interval for the next simulation run. If an entry is invalid he is asked to retry. He then has the opportunity to enter modifications to the input job stream. If his answer is the number 0 (or any other number except 1) no modification is wanted and the program goes on.

If his answer is 1 the user can change the value of any global variable and of any variable within the procedure GENERATE_JOBS. To obtain the variable names the user is referred to the program listing. An example how to change the job arrival rate (variable name: sim_parameters.alpha) is given below. Further details about format and restrictions of data-directed input without data list, as it is used in the simulation program, are given in: IBM PL/I(F) Language Reference Manual, chap. 9, paragraph: Data-Directed Data Specifications. To end the modification mode the user must enter a NULL line (carriage return only).

```
RUN PARAMETERS:
TIME (0 < SEC. < 604801) :
3600

JOBS (0 < NUMBER < 1001) :
100

JOB STREAM MODIFICATIONS? (1=YES, 0=NO)
1

ENTER:
sim_parameters.alpha=1.1
```

When these parameters are entered the program will generate the requested number of jobs. Depending upon this number it will take between a few seconds and several minutes of wall clock time.

E. ENTERING INITIATOR MODIFICATIONS

All entries to start, modify, and stop Initiators have the same format. Each entry consists of two parts: the Initiator number and the associated job classes.

Possible Initiator numbers are 1-15. Entering any other number will cause the programm to end the Initiator modification mode and to go to the next step. To start an Initiator an unused number must be entered. To stop or modify an Initiator its old number has to be entered. Multiple modifications of the same Initiator are valid; the last entry will be used.

Each Initiator can be associated with up to eight job classes. Valid job classes are A-O. A blank character in the input stream is skipped. Entering an invalid job class will cause the Initiator to stop. It is good practice to enter the word 'STOP' (which contains three invalid job classes) to terminate an Initiator. Any input string containing more than eight characters will be truncated. Assigning the same job class more than once to one Initiator is illogical, but it is a valid operation.

If an Initiator is stopped, it is terminated immediately except after restart 4 and restart 6. In these cases the Initiator will finish the current job and then terminate.

Examples:

```
MODIFY INITIATORS:
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
1

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' abc'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
2

ENTER JOB CLASSES 'A-O' OR 'STOP' :
'stop'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
15

ENTER JOB CLASSES 'A-O' OR 'STOP' :
'oabcefk'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
0
```

F. ENTERING TRACE PARAMETERS

Now the user has the choice to select simulation trace, a core map, and statistics gathering in that order by entering a 1 as shown below. If these options are not desired, a number not equal to 1 should be entered.

```
SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :
1 1 1
```

Thereafter the actual simulation is executed. Depending upon the parameters used, the simulation will take between a few seconds and several minutes.

G. ENTERING RESTART PARAMETERS

At the end of one simulation step the user must enter a restart parameter ranging from zero to six:

0 - Stop simulation program

1 - Start new simulation run.
The simulation time is reset to zero and the queues are cleared. The program will start at the beginning allowing all parameters to be entered.

2 - Start new simulation run.
The simulation time is reset to zero and the queues are cleared. The program will start at the beginning again. It will use the run parameters and the input job stream from the previous run. The user can enter only system modifications, Initiator modifications, and trace parameters.

3 - Same as 1.
In addition the seed for the random number generator will be changed at random.

4 - Continue simulation run.
The current simulation time, the status of the queues, and the status of the Initiators are kept. The program will continue from the current point allowing all parameters except system modifications to be entered again.

5 - Continue simulation run.

The current simulation time, the status of the queues, and the status of the Initiators are kept. The program will continue from the current point. It will use the run parameters and the input job stream from the previous run. The user can enter only Initiator modifications and trace parameters.

6 - Same as 4.

In addition the seed for the random number generator will be changed at random.

The job classes associated with the Initiators will be kept in all restart cases.

Since the seed for the random number generator starts with a fixed initial value, the simulation runs are always reproducable when the same parameter entries are used again. However, with restart 3 or restart 6 the seed is changed at random by means of the computer clock. These runs use a truly random seed and cannot be reproduced.

Example:

```
RESTART PARAM. (0 - 6) :
4
```

III. OBTAINING RESULTS

A. OBTAINING SIMULATION TRACE AND CORE MAP

After the simulation program has terminated the simulation trace and the core map are on file FILE P1. This file can be printed directly using the following command:

```
o printcc file p1
R;
```

B. OBTAINING STATISTICAL DATA

Statistical data are kept on file FILE STAT. This file cannot be printed directly; it must be processed by the supporting statistical program. It is assumed that a STA TEXT file is on the user's disk. Then the statistical program can be started with the following commands:

```
filedef sysin con blksize 80
R;
filedef sysprint con
R;
$ sta
EXECUTICN BEGINS...
```

The user is asked to enter those Initiators for which he wants statistical data. SEVEN numbers must be entered. If the user wants statistics for less than seven Initiators, he can fill up the parameter list with trailing zeros. There is no validity check on the Initiator numbers. Trailing zeros will cause blank output; any other invalid number will cause unpredictable program behavior and erroneous results. If statistics for more than seven Initiators are wanted the statistical program must be run a second time.

```
ENTER INITIATORS:
1 2 15 0 0 0 0
```

Next the user is asked to specify the desired types of statistical data. The first three parameters refer to the three different output formats 1-3. Examples of these formats are given in Appendix B. The fourth parameter indicates if the statistics should be added to the summary output. The purpose of this parameter is to accumulate statistics for continuous simulation steps. Entering the number 1 will cause the appropriate format or summary to be written, entering any other number will suppress this function.

Examples:

```
ENTER STATISTICS TYPES:
1 1 1 0

ENTER STATISTICS TYPES:
1 1 1 1

ENTER STATISTICS TYPES:
0 0 0 1
```

The entry of statistics parameters will be repeated for each statistical record on the file.

If the entire file has been read, the user must enter three parameters for the summary report. These parameters refer to the output formats 1-3. Again, entering the number 1 will cause the appopriate format to be written. The summary report should only be requested when statistics have been accumulated from continuous simulation steps. If the steps are non-continuous, the output may contain incorrect results.

```
ENTER SUMMARY TYPES:
1 1 1

R;
```

After the statistical program has terminated the results are on file FILE F1 and can be printed using the following command:

```
o printcc file f1
R;
```

Examples of simulation results are given in Appendix B.

APPENDIX B

DEMONSTRATION RUN

This Appendix contains a demonstration run of the simulation model executed under CP/CMS at the Naval Postgraduate School.

A three hour period is simulated and snapshots are taken after each hour. The use of different input parameters demonstates most of the features of the model. The restriction in system resources (tapes = 1, disks = 0, core = 500 K) is used to demonstrate certain conditions such as job cancellations because of lack of devices, jobs waiting for devices, and Initiators waiting for main memory.

Included in this Appendix are the results of the demonstration run: simulation traces, core maps, and statistical reports.

TABLE OF CONTENTS

I. DEMONSTRATION RUN

A. PREPARATION

```
login 0860p10 500k
ENTER PASSWORD:

********
ENTER 4-DIGIT PROJECT NUMBER FOLLOWED BY ...
0748cr62
FILES:- 02 RDR, NO PRT, NO PUN
READY AT 10.12.16 ON 05/09/77
CMS VERSION 3.2

list
FILENAME FILETYPE MODE NO.REC. DATE
PROFILE  EXEC      P1       1   5/09
SIM      TEXT      P1     177   5/09
STA      TEXT      P1      44   5/09
R;

cp define t2314 192 7
R;

login 192 b
IOERR R 0794 NRF-MADDMK       ADDRESS: 192
** B (192) DEVICE ERROR **
E(00001)

format b all
** "FORMAT B" WILL ERASE ALL YOUR B-DISK (192) FILES **
** DO YOU WISH TO CONTINUE?  ENTER "YES" OR "NO":
yes
ENTER 6-BYTE LABEL (IF WANTED), OR NULL LINE (IF  NOT):

FORMATTING B-DISK (2314)...
B (192): 007 CYL
R;

release 192 b
R;

login 192 p
192  REPLACES P (191)
R;

login 191 b,p
B (191) R/O
R;
```

B. SIMULATION (FIRST HOUR)

```
filedef sysin con blksize 80
R;

filedef sysprint con
R;

$ sim
*EXECUTION BEGINS...

SYSTEM MODIFICATIONS? (1=YES, 0=NO)
1

ENTER:
mod.tapes=1
mod.disks=0
mod.core_h=640
mod.core_l=140

RUN PARAMETERS:
TIME (0 < SEC. < 604801) :
3600

JOBS (0 < NUMBER < 1001) :
100

JOB STREAM MODIFICATIONS? (1=YES, 0=NO)
0

**

MODIFY INITIATORS:
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
1

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' abc'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
2

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' oef'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
15

' koa'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
0

SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :
1 1 1
**
```

C. SIMULATION (SECOND HOUR)

```
RESTART PARAM. (0 - 6) :
4

RUN PARAMETERS:
TIME (0 < SEC. < 604801) :
3600

JOBS (0 < NUMBER < 1001) :
100

JOB STREAM MODIFICATIONS? (1=YES, 0=NO)
1

ENTER:
sim_parameters.alpha=1.1
**

MODIFY INITIATORS:
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
15

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' stop'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
3

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' kef'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
2

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' oa'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
0

SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :
1 1 1
**
```

D. SIMULATION (THIRD HOUR)

```
RESTART PARAM. (0 - 6) :
5

MODIFY INITIATORS:
ENTER N (1-15 = INIT.# , 0 = END MOD.) :
1

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' o'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
2

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' a'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
3

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' c'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
4

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' b'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
5

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' ef'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
15

ENTER JOB CLASSES 'A-O' OR 'STOP' :
' k'

ENTER N (1-15 = INIT.# , 0 = END MOD.) :
0

SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :
1 1 1
**

RESTART PARAM. (0 - 6) :
0

R;
```

E. OBTAINING RESULTS

```
o printcc file p1
R;

filedef sysin con blksize 80
R;

filedef sysprint con
R;

$ sta
EXECUTION BEGINS...

ENTER INITIATORS:
1 2 3 4 5 15 0

ENTER STATISTICS TYPES:
1 1 1 1

ENTER STATISTICS TYPES:
1 1 1 1

ENTER STATISTICS TYPES:
1 1 1 1

ENTER SUMMARY TYPES:
1 1 1

R;


o printcc file f1
R;
```

II. DEMONSTRATION RESULTS

A. CORE MAPS

TIME OF SNAPSHOT: 3600

USAGE OF MAIN MEMORY

HIGH	LOW	CORE	INIT.#	JOB #
640	540	100	1	65
540	390	150	2	80
390	140	250	free	

TIME OF SNAPSHOT: 7200

USAGE OF MAIN MEMORY

HIGH	LOW	CORE	INIT.#	JOB #
640	540	100	2	183
540	440	100	1	154
440	140	300	free	

TIME OF SNAPSHOT: 10800

USAGE OF MAIN MEMORY

HIGH	LOW	CORE	INIT.#	JOB #
640	516	124	free	
516	416	100	1	183
416	236	180	4	31
236	140	96	free	

B. TRACE (FIRST HOUR)

```
*          0 * INIT.     1 WAITING FOR WORK
*          0 * INIT.     2 WAITING FOR WORK
*          0 * INIT.    15 WAITING FOR WORK
*        195 * JOB       1 STARTED BY INITIATOR  1
*        282 * JOB       1 TERMINATED
*        282 * JOB       2 STARTED BY INITIATOR  1
*        339 * JOB       3 STARTED BY INITIATOR 15
*        432 * JOB       7 STARTED BY INITIATOR  2
*        491 * JOB       7 TERMINATED
*        491 * INIT.     2 WAITING FOR WORK
*        655 * JOB      13 STARTED BY INITIATOR  2
*        744 * JOB      13 TERMINATED
*        744 * INIT.     2 WAITING FOR WORK
*        920 * JOB      17 STARTED BY INITIATOR  2
*        996 * JOB      17 TERMINATED
*        996 * JOB       3 TERMINATED
*        996 * INIT.     2 WAITING FOR WORK
*        996 * JOB       4 STARTED BY INITIATOR 15
*       1027 * JOB       4 TERMINATED
*       1027 * JOB       6 STARTED BY INITIATOR 15
*       1062 * JOB       6 TERMINATED
*       1062 * JOB       9 STARTED BY INITIATOR 15
*       1265 * JOB       2 TERMINATED
*       1265 * JOB      10 STARTED BY INITIATOR  1
*       1287 * JOB      10 TERMINATED
*       1287 * JOB      12 STARTED BY INITIATOR  1
*       1333 * JOB       9 TERMINATED
*       1333 * JOB      14 STARTED BY INITIATOR 15
*       1445 * JOB      14 TERMINATED
*       1445 * JOB      21 STARTED BY INITIATOR 15
*       1515 * JOB      27 STARTED BY INITIATOR  2
*       1543 * JOB      12 TERMINATED
*       1543 * JOB      22 STARTED BY INITIATOR  1
*       1574 * JOB      21 TERMINATED
*       1574 * JOB      25 STARTED BY INITIATOR 15
*       1596 * JOB      27 TERMINATED
*       1596 * INIT.     2 WAITING FOR WORK
*       1793 * JOB      22 TERMINATED
*       1793 * JOB      26 STARTED BY INITIATOR  1
*       1818 * JOB      25 TERMINATED
*       1818 * JOB      28 STARTED BY INITIATOR 15
*       1821 * JOB      28 TERMINATED
*       1821 * JOB      29 STARTED BY INITIATOR 15
*       1851 * JOB      26 TERMINATED
*       1851 * JOB      32 STARTED BY INITIATOR  1
*       1928 * JOB      32 TERMINATED
*       1928 * JOB      33 STARTED BY INITIATOR  1
*       1930 * JOB      35 STARTED BY INITIATOR  2
*       2038 * JOB      35 TERMINATED
*       2038 * INIT.     2 WAITING FOR WORK
*       2050 * JOB      33 TERMINATED
*       2050 * JOB      34 STARTED BY INITIATOR  1
*       2133 * JOB      29 TERMINATED
*       2133 * JOB      37 STARTED BY INITIATOR 15
*       2142 * JOB      38 STARTED BY INITIATOR  2
*       2193 * JOB      38 TERMINATED
*       2193 * INIT.     2 WAITING FOR WORK
*       2200 * JOB      40 STARTED BY INITIATOR  2
*       2273 * JOB      37 TERMINATED
*       2273 * JOB      41 STARTED BY INITIATOR 15
*       2369 * JOB      34 TERMINATED
```

```
*      2369 * JOB      39 STARTED BY INITIATOR  1
*      2414 * JOB      41 TERMINATED
*      2414 * JOB      43 STARTED BY INITIATOR 15
*      2448 * JOB      39 TERMINATED
*      2448 * JOB       5 STARTED BY INITIATOR  1
*      2497 * JOB      43 TERMINATED
*      2497 * JOB      47 STARTED BY INITIATOR 15
*      2555 * JOB      47 TERMINATED
*      2555 * JOB      51 STARTED BY INITIATOR 15
*      2636 * JOB      40 TERMINATED
*      2636 * JOB      48 STARTED BY INITIATOR  2
*      2692 * JOB      48 : ALLOCATE 1 DATA SET
*      2739 * JOB      51 TERMINATED
*      2739 * JOB      50 STARTED BY INITIATOR 15
*      2879 * JOB       5 TERMINATED
*      2879 * JOB      54 STARTED BY INITIATOR  1
*      3090 * JOB      48 TERMINATED
*      3090 * JOB      66 STARTED BY INITIATOR  2
*      3141 * JOB      66 TERMINATED
*      3141 * JOB      67 STARTED BY INITIATOR  2
*      3145 * JOB      54 TERMINATED
*      3145 * JOB      55 STARTED BY INITIATOR  1
*      3170 * JOB      55 TERMINATED
*      3170 * JOB      56 STARTED BY INITIATOR  1
*      3227 * JOB      67 TERMINATED
*      3227 * JOB      68 STARTED BY INITIATOR  2
*      3233 * JOB      56 TERMINATED
*      3233 * JOB      61 STARTED BY INITIATOR  1
*      3292 * JOB      61 TERMINATED
*      3292 * JOB      64 STARTED BY INITIATOR  1
*      3327 * JOB      50 TERMINATED
*      3327 * JOB      62 STARTED BY INITIATOR 15
*      3346 * JOB      62 TERMINATED
*      3346 * JOB      63 STARTED BY INITIATOR 15
*      3408 * JOB      68 TERMINATED
*      3408 * JOB      49 STARTED BY INITIATOR  2
*      3431 * JOB      49 TERMINATED
*      3431 * INIT.     2 WAITING FOR WORK
*      3499 * JOB      80 STARTED BY INITIATOR  2
*      3564 * JOB      64 TERMINATED
*      3564 * JOB      65 STARTED BY INITIATOR  1
```

C. TRACE (SECOND HOUR)

```
*      3600 * INIT.    3 WAITING FOR WORK
*      3673 * JOB     80 TERMINATED
*      3673 * JOB     69 STARTED BY INITIATOR  2
*      3719 * JOB     63 : MOUNT  1 TAPE(S)
*      3737 * JOB     65 TERMINATED
*      3737 * JOB     70 STARTED BY INITIATOR  1
*      3856 * JOB     63 ALL VOLUMES MOUNTED
*      3904 * JOB     69 TERMINATED
*      3904 * JOB    101 STARTED BY INITIATOR  2
*      3987 * JOB     70 TERMINATED
*      3987 * JOB     72 STARTED BY INITIATOR  1
*      4008 * JOB    101 TERMINATED
*      4008 * JOB    103 STARTED BY INITIATOR  2
*      4035 * JOB    113 STARTED BY INITIATOR  3
*      4070 * JOB    103 TERMINATED
*      4070 * JOB    109 STARTED BY INITIATOR  2
*      4239 * JOB    109 TERMINATED
*      4239 * JOB    110 STARTED BY INITIATOR  2
*      4264 * JOB    113 TERMINATED
*      4264 * JOB    120 STARTED BY INITIATOR  3
*      4349 * JOB     63 TERMINATED
*      4473 * JOB     72 TERMINATED
*      4473 * JOB     73 STARTED BY INITIATOR  1
*      4536 * JOB     73 TERMINATED
*      4536 * JOB     74 STARTED BY INITIATOR  1
*      4555 * JOB    110 TERMINATED
*      4555 * JOB    111 STARTED BY INITIATOR  2
*      4581 * JOB    111 TERMINATED
*      4581 * JOB    118 STARTED BY INITIATOR  2
*      4653 * JOB     74 TERMINATED
*      4653 * JOB     76 STARTED BY INITIATOR  1
*      4672 * JOB    118 TERMINATED
*      4672 * JOB    119 STARTED BY INITIATOR  2
*      4781 * JOB     76 TERMINATED
*      4781 * JOB    105 STARTED BY INITIATOR  1
*      4819 * JOB    119 TERMINATED
*      4819 * JOB    123 STARTED BY INITIATOR  2
*      4924 * JOB    105 TERMINATED
*      4924 * JOB    107 STARTED BY INITIATOR  1
*      5001 * JOB    123 TERMINATED
*      5001 * JOB    125 STARTED BY INITIATOR  2
*      5116 * JOB    107 TERMINATED
*      5116 * JOB    114 STARTED BY INITIATOR  1
*      5138 * JOB    125 TERMINATED
*      5138 * JOB    129 STARTED BY INITIATOR  2
*      5186 * JOB    129 TERMINATED
*      5186 * JOB    136 STARTED BY INITIATOR  2
*      5376 * JOB    136 TERMINATED
*      5376 * JOB    137 STARTED BY INITIATOR  2
*      5379 * JOB    137 TERMINATED
*      5379 * JOB    139 STARTED BY INITIATOR  2
*      5387 * JOB    139 TERMINATED
*      5387 * JOB    142 STARTED BY INITIATOR  2
*      5494 * JOB    142 TERMINATED
*      5494 * JOB    144 STARTED BY INITIATOR  2
*      5510 * JOB    114 TERMINATED
*      5510 * JOB    115 STARTED BY INITIATOR  1
*      5585 * JOB    115 TERMINATED
*      5585 * JOB    116 STARTED BY INITIATOR  1
*      5589 * JOB    144 TERMINATED
*      5589 * JOB    121 STARTED BY INITIATOR  2
```

```
*      5733 * JOB    116 TERMINATED
*      5733 * JOB    122 STARTED BY  INITIATOR  1
*      5773 * JOB    122 TERMINATED
*      5773 * JOB    126 STARTED BY  INITIATOR  1
*      5924 * JOB    126 TERMINATED
*      5924 * JOB    128 STARTED BY  INITIATOR  1
*      5942 * JOB    128 TERMINATED
*      5942 * JOB    133 STARTED BY  INITIATOR  1
*      6078 * JOB    133 TERMINATED
*      6078 * JOB    138 STARTED BY  INITIATOR  1
*      6134 * JOB    121 TERMINATED
*      6134 * JOB    156 STARTED BY  INITIATOR  2
*      6301 * JOB    156 TERMINATED
*      6301 * JOB    157 STARTED BY  INITIATOR  2
*      6339 * JOB    157 TERMINATED
*      6339 * JOB    141 STARTED BY  INITIATOR  2
*      6353 * JOB    138 TERMINATED
*      6353 * JOB    143 STARTED BY  INITIATOR  1
*      6366 * JOB    141 TERMINATED
*      6366 * JOB    145 STARTED BY  INITIATOR  2
*      6372 * JOB    143 TERMINATED
*      6372 * JOB    146 STARTED BY  INITIATOR  1
*      6609 * JOB    145 TERMINATED
*      6609 * JOB    171 STARTED BY  INITIATOR  2
*      6735 * JOB    171 TERMINATED
*      6735 * JOB    177 STARTED BY  INITIATOR  2
*      6749 * JOB    146 TERMINATED
*      6749 * JOB    147 STARTED BY  INITIATOR  1
*      6772 * JOB    177 TERMINATED
*      6772 * JOB    149 STARTED BY  INITIATOR  2
*      6918 * JOB    147 TERMINATED
*      6918 * JOB    151 STARTED BY  INITIATOR  1
*      6965 * JOB    149 TERMINATED
*      6965 * JOB    181 STARTED BY  INITIATOR  2
*      6980 * JOB    181 TERMINATED
*      6980 * JOB    152 STARTED BY  INITIATOR  2
*      7009 * JOB    151 TERMINATED
*      7009 * JOB    153 STARTED BY  INITIATOR  1
*      7097 * JOB    152 TERMINATED
*      7097 * JOB    183 STARTED BY  INITIATOR  2
*      7156 * JOB    153 : ALLOCATE  1 DATA SET
*      7198 * JOB    153 TERMINATED
*      7198 * JOB    154 STARTED BY  INITIATOR  1
```

D. TRACE (THIRD HOUR)

```
*      7200 * JOB     15 STARTED BY INITIATOR  4
*      7200 * JOB    117 STARTED BY INITIATOR  5
*      7200 * JOB    130 STARTED BY INITIATOR 15
*      7268 * JOB    154 TERMINATED
*      7268 * INIT.    1 WAITING FOR WORK
*      7302 * JOB    101 STARTED BY INITIATOR  1
*      7336 * JOB    183 TERMINATED
*      7336 * JOB    158 STARTED BY INITIATOR  2
*      7336 * JOB    158 : ALLOCATE 1 DATA SET
*      7370 * JOB     15 : MOUNT  1 TAPE(S)
*      7406 * JOB    101 TERMINATED
*      7406 * JOB    103 STARTED BY INITIATOR  1
*      7468 * JOB    103 TERMINATED
*      7468 * INIT.    1 WAITING FOR WORK
*      7484 * JOB    109 STARTED BY INITIATOR  1
*      7653 * JOB    109 TERMINATED
*      7653 * JOB    110 STARTED BY INITIATOR  1
*      7768 * JOB    158 TERMINATED
*      7768 * JOB    159 STARTED BY INITIATOR  2
*      7857 * JOB     15 ALL VOLUMES MOUNTED
*      7882 * JOB     15 TERMINATED
*      7882 * JOB     16 STARTED BY INITIATOR  4
*      7969 * JOB    110 TERMINATED
*      7969 * JOB    111 STARTED BY INITIATOR  1
*      7995 * JOB    111 TERMINATED
*      7995 * JOB    118 STARTED BY INITIATOR  1
*      8031 * JOB    159 TERMINATED
*      8031 * JOB    160 STARTED BY INITIATOR  2
*      8086 * JOB    118 TERMINATED
*      8086 * JOB    119 STARTED BY INITIATOR  1
*      8170 * JOB    160 TERMINATED
*      8170 * JOB    167 STARTED BY INITIATOR  2
*      8182 * JOB    167 TERMINATED
*      8182 * JOB    170 STARTED BY INITIATOR  2
*      8233 * JOB    119 TERMINATED
*      8233 * JOB    123 STARTED BY INITIATOR  1
*      8251 * JOB    170 TERMINATED
*      8251 * JOB    174 STARTED BY INITIATOR  2
*      8400 * JOB    174 TERMINATED
*      8400 * JOB    179 STARTED BY INITIATOR  2
*      8415 * JOB    123 TERMINATED
*      8415 * JOB    125 STARTED BY INITIATOR  1
*      8552 * JOB    125 TERMINATED
*      8552 * JOB    129 STARTED BY INITIATOR  1
*      8600 * JOB    129 TERMINATED
*      8600 * JOB    136 STARTED BY INITIATOR  1
*      8769 * JOB     16 TERMINATED
*      8769 * JOB     18 STARTED BY INITIATOR  4
*      8784 * JOB    179 TERMINATED
*      8784 * JOB    180 STARTED BY INITIATOR  2
*      8790 * JOB    136 TERMINATED
*      8797 * JOB    180 TERMINATED
*      8797 * JOB    137 STARTED BY INITIATOR  1
*      8800 * JOB    137 TERMINATED
*      8800 * JOB    105 STARTED BY INITIATOR  2
*      8876 * JOB     18 TERMINATED
*      8876 * JOB    139 STARTED BY INITIATOR  1
*      8884 * JOB    139 TERMINATED
*      8928 * JOB    117 TERMINATED
*      8928 * JOB    142 STARTED BY INITIATOR  1
*      8928 * JOB     19 STARTED BY INITIATOR  4
```

```
*      8928 * JOB     176 STARTED BY INITIATOR  5
*      8958 * JOB      19 : MOUNT  1 TAPE(S)
*      8989 * JOB     105 TERMINATED
*      8989 * JOB     107 STARTED BY INITIATOR  2
*      9027 * JOB      19 ALL VOLUMES MOUNTED
*      9035 * JOB     142 TERMINATED
*      9035 * JOB     144 STARTED BY INITIATOR  1
*      9130 * JOB     144 TERMINATED
*      9130 * INIT.     1 WAITING FOR WORK
*      9147 * JOB      19 TERMINATED
*      9147 * JOB      20 STARTED BY INITIATOR  4
*      9181 * JOB     107 TERMINATED
*      9191 * JOB     176 TERMINATED
*      9191 * JOB     114 STARTED BY INITIATOR  2
*      9191 * JOB     117 STARTED BY INITIATOR  5
*      9585 * JOB     114 TERMINATED
*      9585 * JOB     115 STARTED BY INITIATOR  2
*      9605 * JOB     156 STARTED BY INITIATOR  1
*      9723 * JOB     117 TERMINATED
*      9723 * JOB     178 STARTED BY INITIATOR  5
*      9772 * JOB     156 TERMINATED
*      9772 * JOB     157 STARTED BY INITIATOR  1
*      9798 * JOB     115 TERMINATED
*      9798 * JOB     116 STARTED BY INITIATOR  2
*      9810 * JOB     157 TERMINATED
*      9810 * INIT.     1 WAITING FOR WORK
*      9901 * JOB      20 TERMINATED
*      9901 * JOB      23 STARTED BY INITIATOR  4
*      9946 * JOB     116 TERMINATED
*      9946 * JOB     121 STARTED BY INITIATOR  2
*      9986 * JOB      23 TERMINATED
*      9986 * JOB      30 STARTED BY INITIATOR  4
*     10149 * JOB     178 WAITING FOR  1 DISK(S)
*     10186 * JOB     171 STARTED BY INITIATOR  1
*     10278 * JOB     178 CANCELLED BY OPERATOR
*     10278 * JOB     178 TERMINATED
*     10312 * JOB     171 TERMINATED
*     10312 * JOB     176 STARTED BY INITIATOR  5
*     10312 * INIT.     1 WAITING FOR WORK
*     10334 * JOB     177 STARTED BY INITIATOR  1
*     10364 * JOB      30 : MOUNT  1 TAPE(S)
*     10371 * JOB     177 TERMINATED
*     10371 * INIT.     1 WAITING FOR WORK
*     10500 * JOB      30 ALL VOLUMES MOUNTED
*     10511 * JOB     181 STARTED BY INITIATOR  1
*     10526 * JOB     181 TERMINATED
*     10526 * INIT.     1 WAITING FOR WORK
*     10630 * JOB     183 STARTED BY INITIATOR  1
*     10632 * JOB      30 TERMINATED
*     10632 * JOB      31 STARTED BY INITIATOR  4
*     10750 * JOB     121 TERMINATED
*     10750 * JOB     122 STARTED BY INITIATOR  2
*     10790 * JOB     122 TERMINATED
*     10790 * JOB     126 STARTED BY INITIATOR  2
```

E. STATISTICS (FIRST HOUR, FORM 1)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 ABC	2 OEF	3 STOP	4 STOP	5 STOP	15 STOP
CLASS A	1:	15	0	0	0	0	0
	2:	0.250	0.000	0.000	0.000	0.000	0.000
CLASS B	1:	1	0	0	0	0	0
	2:	0.017	0.000	0.000	0.000	0.000	0.000
CLASS C	1:	1	0	0	0	0	0
	2:	0.017	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS F	1:	0	2	0	0	0	0
	2:	0.000	0.033	0.000	0.000	0.000	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	0	11	0	0	0	0
	2:	0.000	0.183	0.000	0.000	0.000	0.000

1: NUMBER OF JOBS STARTED (TOTAL)
2: NUMBER OF JOBS STARTED (PER MIN.)

TIME OF SNAPSHOT: 3600

F. STATISTICS (FIRST HOUR, FORM 2)

INITIATORS
(NUMBER AND ASSOCIATED JOB CLASSES)

		1 ABC	2 OEF	3 STOP	4 STOP	5 STOP	15 STOP
TIME_AC	1:	3600	3600	0	0	0	3600
TIME_SJ	1:	3405	1664	0	0	0	2968
	2:	953	436	0	0	0	405
	3:	94.6	46.2	0.0	0.0	0.0	82.4
TIME_WM	1:	0	125	0	0	0	293
	2:	0	125	0	0	0	254
	3:	0.0	3.5	0.0	0.0	0.0	8.1
TIME_WD	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WV	1:	0	7	0	0	0	0
	2:	0	7	0	0	0	0
	3:	0.0	0.2	0.0	0.0	0.0	0.0
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	195	1804	0	0	0	339
	2:	195	519	0	0	0	339
	3:	5.4	50.1	0.0	0.0	0.0	9.4

TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES)
TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
TIME_WM: TIME WAITING FOR MAIN MEMORY
TIME_WD: TIME WAITING FOR DEVICE(S)
TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
2: MAXIMUM TIME IN SEC.
3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 3600

G. STATISTICS (FIRST HOUR, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	32	26	81.3	748	333	235
CLASS B	20	1	5.0	2087	2087	--
CLASS C	12	1	8.3	50	50	--
CLASS D	0	0	--	--	--	--
CLASS E	0	0	--	--	--	--
CLASS F	2	2	100.0	1015	653	513
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	2	2	100.0	558	550	11
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	15	15	100.0	212	58	78

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
J.A. # = JOBS STARTED (TOTAL NUMBER)
J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 3600

H. STATISTICS (SECOND HOUR, FORM 1)

INITIATORS
(NUMBER AND ASSOCIATED JOB CLASSES)

		1 ABC	2 OA	3 KEF	4 STOP	5 STOP	15 STOP
CLASS A	1:	21	6	0	0	0	0
	2:	0.350	0.100	0.000	0.000	0.000	0.000
CLASS B	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS C	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS F	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	2	0	0	0
	2:	0.000	0.000	0.033	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	0	21	0	0	0	0
	2:	0.000	0.350	0.000	0.000	0.000	0.000

1: NUMBER OF JOBS STARTED (TOTAL)
2: NUMBER OF JOBS STARTED (PER MIN.)

TIME OF SNAPSHOT: 7200

I. STATISTICS (SECOND HOUR, FORM 2)

INITIATORS
(NUMBER AND ASSOCIATED JOB CLASSES)

		1 ABC	2 OA	3 KEF	4 STOP	5 STOP	15 STOP
TIME_AC	1:	3600	3600	3600	0	0	749
TIME_SJ	1:	3506	3536	229	0	0	437
	2:	326	439	229	0	0	208
	3:	97.4	98.2	6.4	0.0	0.0	58.3
TIME_WM	1:	86	64	2936	0	0	75
	2:	86	64	2936	0	0	102
	3:	2.4	1.8	81.6	0.0	0.0	23.4
TIME_WD	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WV	1:	8	0	0	0	0	137
	2:	8	0	0	0	0	137
	3:	0.2	0.0	0.0	0.0	0.0	18.3
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	0	0	435	0	0	0
	2:	0	0	435	0	0	0
	3:	0.0	0.0	12.1	0.0	0.0	0.0

TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES)
TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
TIME_WM: TIME WAITING FOR MAIN MEMORY
TIME_WD: TIME WAITING FOR DEVICE(S)
TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
2: MAXIMUM TIME IN SEC.
3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 7200

J. STATISTICS (SECOND HOUR, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	35	27	77.1	1479	1224	232
CLASS B	30	0	--	--	--	--
CLASS C	23	0	--	--	--	--
CLASS D	0	0	--	--	--	--
CLASS E	2	0	--	--	--	--
CLASS F	1	0	--	--	--	--
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	8	2	25.0	13	7	9
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	21	21	100.0	651	323	206

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
J.A. # = JOBS STARTED (TOTAL NUMBER)
J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 7200

K. STATISTICS (THIRD HOUR, FORM 1)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 O	2 A	3 C	4 B	5 EF	15 STOP
CLASS A	1:	0	16	0	0	0	0
	2:	0.000	0.267	0.000	0.000	0.000	0.000
CLASS B	1:	0	0	0	8	0	0
	2:	0.000	0.000	0.000	0.133	0.000	0.000
CLASS C	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	4	0
	2:	0.000	0.000	0.000	0.000	0.067	0.000
CLASS F	1:	0	0	0	0	1	0
	2:	0.000	0.000	0.000	0.000	0.017	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	21	0	0	0	0	0
	2:	0.350	0.000	0.000	0.000	0.000	0.000

1: NUMBER OF JOBS STARTED (TOTAL)
2: NUMBER OF JOBS STARTED (PER MIN.)

TIME OF SNAPSHOT: 10800

L. STATISTICS (THIRD HOUR, FORM 2)

INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)

		1 O	2 A	3 C	4 B	5 EF	15 STOP
TIME_AC	1:	3600	3600	3600	3600	3600	3600
TIME_SJ	1:	2306	3120	0	2733	750	9
	2:	316	439	0	462	203	9
	3:	64.1	86.7	0.0	75.9	20.8	0.3
TIME_WM	1:	127	466	3600	175	2721	3591
	2:	76	259	3600	69	1571	3591
	3:	3.5	12.9	100.0	4.9	75.6	99.8
TIME_WD	1:	0	0	0	0	129	0
	2:	0	0	0	0	129	0
	3:	0.0	0.0	0.0	0.0	3.6	0.0
TIME_WV	1:	0	14	0	692	0	0
	2:	0	14	0	487	0	0
	3:	0.0	0.4	0.0	19.2	0.0	0.0
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	1167	0	0	0	0	0
	2:	475	0	0	0	0	0
	3:	32.4	0.0	0.0	0.0	0.0	0.0

TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES)
TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
TIME_WM: TIME WAITING FOR MAIN MEMORY
TIME_WD: TIME WAITING FOR DEVICE(S)
TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
2: MAXIMUM TIME IN SEC.
3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 10800

M. STATISTICS (THIRD HOUR, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	37	16	43.2	2814	1850	439
CLASS B	41	8	19.5	8884	7830	794
CLASS C	35	0	--	--	--	--
CLASS D	0	0	--	--	--	--
CLASS E	4	4	100.0	3084	1697	1305
CLASS F	2	1	50.0	2962	2962	--
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	14	1	7.1	2711	2711	--
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	21	21	100.0	465	185	167

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
J.A. # = JOBS STARTED (TOTAL NUMBER)
J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 10800

N. STATISTICS (SUMMARY, FORM 1)

INITIATORS
(NUMBER AND ASSOCIATED JOB CLASSES)

		1 O	2 A	3 C	4 B	5 EF	15 STOP
CLASS A	1:	36	22	0	0	0	0
	2:	0.200	0.122	0.000	0.000	0.000	0.000
CLASS B	1:	1	0	0	8	0	0
	2:	0.006	0.000	0.000	0.133	0.000	0.000
CLASS C	1:	1	0	0	0	0	0
	2:	0.006	0.000	0.000	0.000	0.000	0.000
CLASS D	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS E	1:	0	0	0	0	4	0
	2:	0.000	0.000	0.000	0.000	0.067	0.000
CLASS F	1:	0	2	0	0	1	0
	2:	0.000	0.011	0.000	0.000	0.017	0.000
CLASS G	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS H	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS J	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS K	1:	0	0	2	0	0	0
	2:	0.000	0.000	0.017	0.000	0.000	0.000
CLASS L	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS M	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS N	1:	0	0	0	0	0	0
	2:	0.000	0.000	0.000	0.000	0.000	0.000
CLASS O	1:	21	32	0	0	0	0
	2:	0.117	0.178	0.000	0.000	0.000	0.000

1: NUMBER OF JOBS STARTED (TOTAL)
2: NUMBER OF JOBS STARTED (PER MIN.)

TIME OF SNAPSHOT: 10800

O. STATISTICS (SUMMARY, FORM 2)

		INITIATORS (NUMBER AND ASSOCIATED JOB CLASSES)					
		1 O	2 A	3 C	4 B	5 EF	15 STOP
TIME_AC	1:	10800	10800	7200	3600	3600	7949
TIME_SJ	1:	9217	8320	229	2733	750	3414
	2:	953	439	229	462	203	405
	3:	85.3	77.0	3.2	75.9	20.8	42.9
TIME_WM	1:	213	655	6536	175	2721	4059
	2:	86	259	3600	69	1571	3591
	3:	2.0	6.1	90.8	4.9	75.6	51.1
TIME_WD	1:	0	0	0	0	129	0
	2:	0	0	0	0	129	0
	3:	0.0	0.0	0.0	0.0	3.6	0.0
TIME_WV	1:	8	21	0	692	0	137
	2:	8	14	0	487	0	137
	3:	0.1	0.2	0.0	19.2	0.0	1.7
TIME_WS	1:	0	0	0	0	0	0
	2:	0	0	0	0	0	0
	3:	0.0	0.0	0.0	0.0	0.0	0.0
TIME_WW	1:	1362	1804	435	0	0	339
	2:	475	519	435	0	0	339
	3:	12.6	16.7	6.0	0.0	0.0	4.3

TIME_AC: TIME ACTIVE (=TIME_SJ + SUM OF ALL WAITING TIMES)
TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)
TIME_WM: TIME WAITING FOR MAIN MEMORY
TIME_WD: TIME WAITING FOR DEVICE(S)
TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED
TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE
TIME_WW: TIME WAITING FOR WORK

1: TOTAL TIME IN SEC.
2: MAXIMUM TIME IN SEC.
3: TIME IN % OF ACTIVE TIME

TIME OF SNAPSHOT: 10800

P. STATISTICS (SUMMARY, FORM 3)

	J.A. #	J.S. #	J.S. %	MAX. W	MEAN W	DEV. W
CLASS A	90	69	76.7	2814	1033	665
CLASS B	42	9	21.4	8884	7192	2054
CLASS C	36	1	2.8	50	50	--
CLASS D	0	0	--	--	--	--
CLASS E	4	4	100.0	3084	1697	1305
CLASS F	4	3	75.0	2962	1422	1382
CLASS G	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS H	0	0	--	--	--	--
CLASS J	0	0	--	--	--	--
CLASS K	18	5	27.8	2711	765	1121
CLASS L	0	0	--	--	--	--
CLASS M	0	0	--	--	--	--
CLASS N	0	0	--	--	--	--
CLASS O	57	57	100.0	651	202	194

J.A. # = JOBS AVAILABLE (TOTAL NUMBER)
J.A. # = JOBS STARTED (TOTAL NUMBER)
J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)
MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)
MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)
DEV. W = STANDARD DEVIATION OF WAITING TIME

TIME OF SNAPSHOT: 10800

APPENDIX C

COMPUTER PROGRAMS

```
SIM:    PROCEDURE OPTIONS (MAIN);

        /* GLOBAL VARIABLES */

        DCL   STAT FILE RECORD SEQUENTIAL OUTPUT
              ENVIRONMENT (F(1744) CONSECUTIVE);

        DCL   P1 FILE PRINT;

        DCL   1 SYSTEM,
              2 USED_CORE(18),
              3 HIGH           FIXED BIN,
              3 LOW            FIXED BIN,
              3 INIT           FIXED BIN,
              2 FREE_CORE(34)FIXED BIN,
              2 D_A_SPACE      FIXED BIN(31),
              2 IN_SPOOL       FIXED BIN(31),
              2 DISKS          FIXED BIN,
              2 TAPES          FIXED BIN,
              2 INITIAL_VALUES,
              3 I_IN_SPOOL     FIXED BIN(31) INIT ( 45000 ),
              3 I_D_A_SPACE    FIXED BIN(31) INIT ( 100 ),
              3 I_CORE_HIGH    FIXED BIN INIT ( 1140 ),
              3 I_CORE_LOW     FIXED BIN INIT ( 140 ),
              3 I_DISKS        FIXED BIN INIT ( 3 ),
              3 I_TAPES        FIXED BIN INIT ( 9 );

        DCL   1 SIM_PARAMETERS,
              2 NEXT_EOJ       FIXED BIN(31) INIT ( 604801 ),
              2 DELTA_TIME     FIXED BIN(31),
              2 SEED_1         FIXED BIN(31),
              2 SEED_2         FIXED BIN(31),
              2 MIN_CORE       FIXED BIN INIT ( 62 ),
              2 JOB_NUMBER     FIXED BIN INIT ( 1 ),
              2 RESTART        FIXED BIN INIT ( 1 ),
              2 RESULTS        FIXED BIN,
              2 CORE_MAP       FIXED BIN,
              2 TRACE          FIXED BIN,
              2 MAX_JOBS       FIXED BIN,
              2 ALPHA          FLOAT BIN;
```

```
DCL   1 INPUT_STREAM,                                                   SIM00420
        2 INPUT(5000)    LIKE   JOB,                                    SIM00430
        2 JOBS(1000)     FIXED BIN,                                     SIM00440
        2 STEP_COUNT     FIXED BIN,                                     SIM00450
        2 JOB_COUNT      FIXED BIN;                                     SIM00460
                                                                        SIM00470
DCL   1 JOB      BASED (JOB_PTR),                                       SIM00480
        2 ARRIVAL        FIXED BIN(31),                                 SIM00490
        2 DEL_ARRIVAL    FIXED BIN,                                     SIM00500
        2 PRIORITY       FIXED BIN,                                     SIM00510
        2 NUMBER         FIXED BIN,                                     SIM00520
        2 STEPS          FIXED BIN,                                     SIM00530
        2 CLASS          FIXED BIN,                                     SIM00540
        2 CARDS          FIXED BIN;                                     SIM00550
                                                                        SIM00560
DCL   1 STEP    BASED (STEP_PTR),                                       SIM00570
        2 RUN_TIME       FIXED BIN(31),                                 SIM00580
        2 NUMBER         FIXED BIN,                                     SIM00590
        2 CORE_SIZE      FIXED BIN,                                     SIM00600
        2 DISKS          FIXED BIN,                                     SIM00610
        2 TAPES          FIXED BIN,                                     SIM00620
        2 OP_REQUESTS    FIXED BIN,                                     SIM00630
        2 D_A_SPACE      FIXED BIN;                                     SIM00640
                                                                        SIM00650
DCL   1 JOB_QUEUE,                                                      SIM00660
        2 SIZE           FIXED BIN INIT ( 5000 ),                       SIM00670
        2 FREE_HEAD      FIXED BIN,                                     SIM00680
        2 AMOUNT(15)     FIXED BIN,                                     SIM00690
        2 HEAD(15)       FIXED BIN,                                     SIM00700
        2 LINK(5000),                                                   SIM00710
        3 JOB_LINK       FIXED BIN,                                     SIM00720
        3 STEP_LINK      FIXED BIN,                                     SIM00730
        2 QUEUE(5000)    LIKE   JOB;                                    SIM00740
                                                                        SIM00750
DCL   1 TIME,                                                           SIM00760
        2 SYSTEM         FIXED BIN(31),                                 SIM00770
        2 STOP           FIXED BIN(31),                                 SIM00780
        2 READER         FIXED BIN(31),                                 SIM00790
        2 INITIATOR(15)FIXED BIN(31);                                   SIM00800
```

```
DCL  1 IN(15),                                            SIM00820
      2 STATUS          FIXED BIN,                        SIM00830
      2 JOB_NO          FIXED BIN,                        SIM00840
      2 JOB_INDEX       FIXED BIN,                        SIM00850
      2 STEP_INDEX      FIXED BIN,                        SIM00860
      2 STOP_SWITCH     FIXED BIN,                        SIM00870
      2 CLASS(8)        FIXED BIN,                        SIM00880
      2 RESOURCES,                                        SIM00890
       3 D_A_SPACE      FIXED BIN,                        SIM00900
       3 CORE_HIGH      FIXED BIN,                        SIM00910
       3 CORE_LOW       FIXED BIN,                        SIM00920
       3 DISKS          FIXED BIN,                        SIM00930
       3 TAPES          FIXED BIN,                        SIM00940
       3 CARDS          FIXED BIN;                        SIM00950
                                                          SIM00960
DCL  1 STATISTICS,                                        SIM00970
      2 TIME_LC(15)     FIXED BIN(31),                    SIM00980
      2 TIME_SNAP       FIXED BIN(31),                    SIM00990
      2 MAXI(15),                                         SIM01000
       3 JOB_WAITM      FIXED BIN(31),                    SIM01010
       3 MAX_SJ         FIXED BIN(31),                    SIM01020
       3 MAX_WD         FIXED BIN(31),                    SIM01030
       3 MAX_WM         FIXED BIN(31),                    SIM01040
       3 MAX_WS         FIXED BIN(31),                    SIM01050
       3 MAX_WV         FIXED BIN(31),                    SIM01060
       3 MAX_WW         FIXED BIN(31),                    SIM01070
      2 NO(15),                                           SIM01080
       3 TIME_SJ        FIXED BIN(31),                    SIM01090
       3 TIME_WM        FIXED BIN(31),                    SIM01100
       3 TIME_WD        FIXED BIN(31),                    SIM01110
       3 TIME_WV        FIXED BIN(31),                    SIM01120
       3 TIME_WS        FIXED BIN(31),                    SIM01130
       3 TIME_WW        FIXED BIN(31),                    SIM01140
       3 JOB_WAIT       FIXED BIN(31),                    SIM01150
       3 JOB_WAITS      FLOAT BIN,                        SIM01160
       3 JOB_START(15)FIXED BIN,                          SIM01170
       3 IN_QUEUE       FIXED BIN,                        SIM01180
       3 ABEND_DEVICE FIXED BIN,                          SIM01190
       3 ABEND_D_SET    FIXED BIN,                        SIM01200
       3 ABEND_SPACE    FIXED BIN,                        SIM01210
       3 OP_ANSWER      FIXED BIN,                        SIM01220
       3 OP_TIME        FIXED BIN,                        SIM01230
       3 JOB_IN         FIXED BIN,                        SIM01240
      2 CLASSES(15)     CHAR   (8);                       SIM01250
```

```
GENERATE_JOBS:    PROCEDURE;
        /* GENERATE INPUT JOB STREAM */

        DCL   SPLIT(15) LABEL
              INIT (      CLASS_A, CLASS_B, CLASS_C, CLASS_K, CLASS_E,
                          CLASS_F, CLASS_K, CLASS_K, CLASS_K, CLASS_K,
                          CLASS_K, CLASS_K, CLASS_K, CLASS_K, CLASS_C);

        DCL   CLASSES(15) INIT(    .3664, .5832, .7086, .7086, .7200,
                                   .7310, .7825, .7825, .7825, .7825,
                                   .7825, .7825, .7825, .7825, 1.00);

        DCL   DEVICES_1(10) INIT( .8887, .9283, .9302, .9321, .9326,
                                  .9329, .9332, .9337, .9921, 1.00);

        DCL   DEVICES_2(11) INIT( .8570, .8951, .8970, .8988, .8993,
                                  .8996, .8999, .9004, .9567, .9643,
                                  1.00);

        DCL   DISK(11) FIXED BIN INIT( 0,1,2,1,1,2,3,3,0,0,0 );
        DCL   TAPE(11) FIXED BIN INIT( 0,0,0,1,2,2,1,2,1,2,3 );

        DCL   REQUESTS(2)     INIT ( .9625, 1.0);
        DCL   RANDOM_X;


REAL:   PROCEDURE (TABLE);

        DCL   TABLE(60), NUMBER, VALUE, X_LOW;

        CALL RANDU (SEED_1,SEED_1,RANDOM_X);
        NUMBER = 1;
        DO WHILE (RANDOM_X > TABLE(NUMBER));
              NUMBER = NUMBER + 1;
        END;
        IF NUMBER  = 1 THEN X_LOW = 0;
        ELSE X_LOW = TABLE (NUMBER - 1);
        VALUE = NUMBER-1+(RANDOM_X-X_LOW)/(TABLE(NUMBER)-X_LOW);
        RETURN (VALUE);

        END REAL;
```

```
INTEGER:      PRCCEDURE (TABLE);

        DCL   TABLE(60), NUMBER;

        CALL RANDU (SEED_1,SEED_1,RANDOM_X);
        NUMBER = 1;
        DO WHILE (RANDOM_X > TABLE(NUMBER));
              NUMBER = NUMBER + 1;
        END;
        RETURN (NUMBER);

        END INTEGER;



SET_DEL_ARRIVAL: PROCEDURE;

        CALL RANDU(SEED_1,SEED_1,RANDCM_X);
        RANDOM_X = - LOG(RANDOM_X) / ALPHA * 60;
        RETURN(RANDOM_X);

        END SET_DEL_ARRIVAL;



        /* SET JOB PARAMETERS */
        PUT SKIP LIST ('JOB STREAM MODIFICATIONS? (1=YES, 0=NO)');
        PUT SKIP;
        GET LIST (INDEX);
        IF INDEX=1
              THEN
              DO;
              PUT SKIP LIST ('ENTER:');
              PUT SKIP;
              GET DATA;
              END;
        INDEX = 1;
        DO I=1 TO MAX_JOBS;
              JOBS(I)          = INDEX;
              JOB_PTR          = ADDR(INPUT(INDEX));
              INDEX            = INDEX + 1;
              JOB.DEL_ARRIVAL  = SET_DEL_ARRIVAL + .5;
              JOB.CLASS        = INTEGER(CLASSES);
              JOB.PRIORITY     = 1;
              JCB.ARRIVAL      = 0;
              JOB.NUMBER       = JOB_NUMBER;
              JCB_NUMBER       = JOB_NUMBER + 1;
              GOTO SPLIT(JOB.CLASS);
```

```
CLASS_A:;
        DCL   STEPS_A(11) INIT(    .4180, .4711, .9780, .9866, .9989,
                                   .9995, .9995, .9995, .9998, .9999,
                                   1.00);

        DCL   CARDS_A(30) INIT(    .3385, .5071, .5878, .6565, .7158,
                                   .7775, .8190, .8497, .8744, .8988,
                                   .9190, .9297, .9400, .9463, .9518,
                                   .9547, .9587, .9630, .9655, .9706,
                                   .9764, .9787, .9827, .9870, .9906,
                                   .9933, .9960, .9978, .9995, 1.00);

        DCL   CORE_A(13) INIT(     .3074, .3118, .3149, .3228, .8267,
                                   .8322, .8421, .8483, .8485, .9618,
                                   .9659, .9704, 1.00);

        DCL   TIME_A(44) INIT(     .0621, .1251, .2230, .3358, .4406,
                                   .5349, .6139, .6734, .7245, .7682,
                                   .8049, .8356, .8603, .8813, .8983,
                                   .9115, .9238, .9334, .9423, .9487,
                                   .9550, .9601, .9648, .9684, .9711,
                                   .9743, .9771, .9794, .9815, .9833,
                                   .9852, .9868, .9884, .9899, .9912,
                                   .9929, .9941, .9950, .9963, .9975,
                                   .9980, .9990, .9996, 1.00);

        JOB.STEPS = INTEGER(STEPS_A);
        JOB.CARDS = REAL(CARDS_A) * 40 + .5;
        DO J=1 TO JOB.STEPS;
              STEP_PTR           = ADDR(INPUT(INDEX));
              INDEX              = INDEX + 1;
              STEP.RUN_TIME      = REAL(TIME_A) * 10 + .5;
              STEP.NUMBER        = J;
              STEP.CORE_SIZE     = INTEGER(CORE_A) * 10 + 50;
              STEP.TAPES         = 0;
              STEP.DISKS         = 0;
              STEP.OP_REQUESTS   = INTEGER(REQUESTS) - 1;
              STEP.D_A_SPACE     = 0;
              IF STEP.CORE_SIZE  = 60 THEN STEP.CCRE_SIZE = MIN_CORE;
        END;
        GOTC LOOP;
```

```
CLASS_B:;
        DCL  STEPS_B(10) INIT(   .3226, .4104, .8871, .9238, .9755,
                                 .9971, .9996, .9998, .9998, 1.00);

        DCL  CARDS_B(30) INIT(   .4004, .5275, .6002, .6469, .6955,
                                 .7364, .7781, .8029, .8209, .8516,
                                 .8651, .8754, .8882, .9006, .9092,
                                 .9180, .9235, .9321, .9418, .9489,
                                 .9546, .9578, .9613, .9664, .9704,
                                 .9750, .9809, .9874, .9916, 1.00);

        DCL  CORE_B(13) INIT(    .2049, .2056, .2258, .2327, .7194,
                                 .7403, .7639, .7718, .7725, .9153,
                                 .9239, .9377, 1.00);

        DCL  TIME_B(40) INIT(    .0962, .2266, .3625, .4835, .5798,
                                 .6572, .7121, .7566, .7937, .8235,
                                 .8478, .8659, .8823, .8966, .9090,
                                 .9182, .9272, .9340, .9398, .9459,
                                 .9512, .9559, .9609, .9649, .9681,
                                 .9715, .9751, .9782, .9807, .9835,
                                 .9853, .9874, .9892, .9913, .9927,
                                 .9945, .9959, .9973, .9986, 1.00);

        JOB.STEPS = INTEGER(STEPS_B);
        JOB.CARDS = REAL(CARDS_B) * 40 + .5;
        DO J=1 TO JOB.STEPS;
             STEP_PTR            = ADDR(INPUT(INDEX));
             INDEX               = INDEX + 1;
             STEP.RUN_TIME       = REAL(TIME_B) * 20 + .5;
             STEP.NUMBER         = J;
             STEP.CORE_SIZE      = INTEGER(CORE_B) * 10 + 50;
             NUM                 = INTEGER(DEVICES_1);
             STEP.TAPES          = TAPE(NUM);
             STEP.DISKS          = DISK(NUM);
             STEP.OP_REQUESTS    = INTEGER(REQUESTS) - 1;
             STEP.D_A_SPACE      = 0;
             IF STEP.CORE_SIZE = 60 THEN STEP.CORE_SIZE = MIN_CORE;
        END;
        GOTO LOOP;
```

```
CLASS_C:;
        DCL   STEPS_C(11) INIT(    .5420, .6035, .8912, .9197, .9816,
                                   .9988, .9994, .9994, .9994, .9997,
                                   1.00);

        DCL   CARDS_C(30) INIT(    .4205, .5851, .6603, .7262, .7672,
                                   .7904, .8066, .8447, .8556, .8669,
                                   .8785, .9036, .9113, .9199, .9272,
                                   .9341, .9424, .9460, .9477, .9500,
                                   .9533, .9576, .9632, .9662, .9695,
                                   .9772, .9901, .9974, .9974, 1.00);

        DCL   CORE_C(11) INIT(     .1516, .1563, .4845, .5063, .5104,
                                   .6434, .6767, .7994, .8150, .8481,
                                   1.00);

        DCL   TIME_C(38) INIT(     .1922, .4056, .5415, .6340, .6919,
                                   .7373, .7730, .8019, .8285, .8479,
                                   .8681, .8826, .8976, .9098, .9196,
                                   .9274, .9343, .9410, .9471, .9528,
                                   .9578, .9625, .9676, .9734, .9771,
                                   .9800, .9826, .9849, .9870, .9879,
                                   .9893, .9905, .9922, .9943, .9959,
                                   .9968, .9985, 1.00);

        JOB.STEPS = INTEGER(STEPS_C);
        JOB.CARDS = REAL(CARDS_C) * 40 + .5;
        DO J=1 TO JOB.STEPS;
              STEP_PTR            = ADDR(INPUT(INDEX));
              INDEX               = INDEX + 1;
              STEP.RUN_TIME       = REAL(TIME_C) * 40 + .5;
              STEP.NUMBER         = J;
              STEP.CORE_SIZE      = INTEGER(CORE_C) * 20 + 40;
              NUM                 = INTEGER(DEVICES_1);
              STEP.TAPES          = TAPE(NUM);
              STEP.DISKS          = DISK(NUM);
              STEP.OP_REQUESTS    = INTEGER(REQUESTS) - 1;
              STEP.D_A_SPACE      = 0;
              IF STEP.CORE_SIZE   = 60 THEN STEP.CORE_SIZE = MIN_CORE;
              IF STEP.CORE_SIZE   = 260 THEN STEP.CORE_SIZE = 250;
        END;
        GOTO LOCP;
```

```
CLASS_E:;
        DCL   STEPS_E( 9) INIT(    .3655, .5241, .9724, .9827, .9930,
                                   .9965, .9965, .9965, 1.00);

        DCL   CARDS_E(30) INIT(    .2769, .5385, .6231, .6538, .7115,
                                   .7962, .8385, .8846, .9077, .9192,
                                   .9192, .9385, .9385, .9500, .9538,
                                   .9615, .9615, .9615, .9654, .9731,
                                   .9846, .9923, .9923, .9962, .9962,
                                   .9962, .9962, .9962, 1.00);

        DCL   CORE_E( 9) INIT(     .1238, .4492, .4524, .4857, .5174,
                                   .5730, .8032, .8302, 1.00);

        DCL   TIME_E(28) INIT(     .2107, .4700, .6013, .6791, .7391,
                                   .7812, .8088, .8363, .8541, .8736,
                                   .8865, .8995, .9125, .9254, .9335,
                                   .9400, .9481, .9530, .9611, .9643,
                                   .9724, .9757, .9821, .9870, .9802,
                                   .9951, .9983, 1.00);

        JOB.STEPS = INTEGER(STEPS_E);
        JOB.CARDS = REAL(CARDS_E) * 80 + .5;
        DO J=1 TO JOB.STEPS;
             STEP_PTR           = ADDR(INPUT(INDEX));
             INDEX              = INDEX + 1;
             STEP.RUN_TIME      = REAL(TIME_E) * 40 + .5;
             STEP.NUMBER        = J;
             STEP.CORE_SIZE     = INTEGER(CORE_E) * 40 + 20;
             NUM                = INTEGER(DEVICES_1);
             STEP.TAPES         = TAPE(NUM);
             STEP.DISKS         = DISK(NUM);
             STEP.OP_REQUESTS   = INTEGER(REQUESTS) - 1;
             STEP.D_A_SPACE     = 0;
             IF STEP.CORE_SIZE = 60 THEN STEP.CORE_SIZE = MIN_CORE;
             IF STEP.CORE_SIZE = 380 THEN STEP.CORE_SIZE = 350;
        END;
        GOTC LOOP;
```

```
CLASS_F:;
        DCL   STEPS_F( 9) INIT(    .1352, .1921, .8042, .8149, .9679,
                                   .9928, .9928, .9964, 1.00);

        DCL   CARDS_F(30) INIT(    .1558, .3116, .3841, .4457, .4819,
                                   .5399, .5616, .5833, .6123, .6522,
                                   .7681, .7899, .8080, .8370, .8659,
                                   .8986, .9058, .9457, .9457, .9674,
                                   1.00);

        DCL   CORE_F(10) INIT(     .1688, .5775, .6096, .8128, .8277,
                                   .8977, .9516, .9516, .9745, 1.00);

        DCL   TIME_F(30) INIT(     .3698, .5815, .6813, .7238, .7640,
                                   .7920, .8090, .8285, .8382, .8540,
                                   .8662, .8808, .8954, .9100, .9185,
                                   .9307, .9428, .9538, .9599, .9635,
                                   .9696, .9732, .9793, .9842, .9891,
                                   .9903, .9903, .9939, .9976, 1.00);

        JOB.STEPS = INTEGER(STEPS_F);
        JOB.CARDS = REAL(CARDS_F) * 80 + .5;
        DO J=1 TO JOB.STEPS;
              STEP_PTR            = ADDR(INPUT(INDEX));
              INDEX               = INDEX + 1;
              STEP.RUN_TIME       = REAL(TIME_F) * 40 + .5;
              STEP.NUMBER         = J;
              STEP.CORE_SIZE      = INTEGER(CORE_F) * 40 + 20;
              NUM                 = INTEGER(DEVICES_1);
              STEP.TAPES          = 0;
              STEP.DISKS          = DISK(NUM);
              STEP.OP_REQUESTS    = INTEGER(REQUESTS) - 1;
              STEP.D_A_SPACE      = 0;
              IF STEP.CORE_SIZE = 60 THEN STEP.CORE_SIZE = MIN_CORE;
              IF STEP.CORE_SIZE = 420 THEN STEP.CORE_SIZE = 400;
        END;
        GOTO LOOP;
```

```
CLASS_K:;
        DCL   STEPS_K(15) INIT(    .3934, .4779, .8934, .9284, .9755,
                                   .9907, .9953, .9961, .9976, .9976,
                                   .9984, .9992, .9992, .9992, 1.00);

        DCL   CARDS_K(30) INIT(    .5362, .6292, .7146, .7577, .8000,
                                   .8377, .8546, .8646, .8715, .8885,
                                   .9069, .9177, .9262, .9346, .9446,
                                   .9556, .9577, .9615, .9654, .9731,
                                   .9800, .9831, .9877, .9923, .9946,
                                   .9954, .9969, .9977, 1.00);

        DCL   CORE_K(20) INIT(     .1851, .5829, .6330, .7929, .8394,
                                   .9027, .9282, .9311, .9466, .9615,
                                   .9696, .9728, .9728, .9903, .9964,
                                   1.00);

        DCL   TIME_K(30) INIT(     .4677, .6357, .7067, .7579, .7936,
                                   .8165, .8316, .8657, .8808, .8946,
                                   .9067, .9145, .9232, .9323, .9407,
                                   .9465, .9529, .9582, .9623, .9677,
                                   .9710, .9741, .9781, .9828, .9859,
                                   .9882, .9909, .9946, .9966, 1.00);

        JOB.CLASS = 11;
        JOB.STEPS = INTEGER(STEPS_K);
        JOB.CARDS = REAL(CARDS_K) * 80 + .5;
        DO J=1 TO JOB.STEPS;
            STEP_PTR            = ADDR(INPUT(INDEX));
            INDEX               = INDEX + 1;
            STEP.RUN_TIME       = REAL(TIME_K) * 40 + .5;
            STEP.NUMBER         = J;
            STEP.CORE_SIZE      = INTEGER(CORE_K) * 40 + 20;
            NUM                 = INTEGER(DEVICES_2);
            STEP.TAPES          = TAPE(NUM);
            STEP.DISKS          = DISK(NUM);
            STEP.OP_REQUESTS    = INTEGER(REQUESTS) - 1;
            STEP.D_A_SPACE      = 0;
            IF STEP.CORE_SIZE   = 60 THEN STEP.CCRE_SIZE = MIN_CORE;
        END;
        GOTO LOOP;
```

```
CLASS_C:;                                                                SIM04550
        DCL   STEPS_O( 2) INIT(    .5524, 1.00);                         SIM04560
                                                                         SIM04570
        JOB.STEPS = INTEGER(STEPS_O);                                    SIM04580
        JOB.CARDS = REAL(CARDS_A) * 40 + .5;                             SIM04590
        DO J=1 TO JOB.STEPS;                                             SIM04600
              STEP_PTR            = ADDR(INPUT(INDEX));                  SIM04610
              INDEX               = INDEX + 1;                           SIM04620
              STEP.RUN_TIME       = REAL(TIME_A) * 10 + .5;              SIM04630
              STEP.NUMBER         = J;                                   SIM04640
              STEP.CORE_SIZE      = INTEGER(CORE_A) * 10 + 50;           SIM04650
              STEP.TAPES          = 0;                                   SIM04660
              STEP.DISKS          = 0;                                   SIM04670
              STEP.OP_REQUESTS    = 0;                                   SIM04680
              STEP.D_A_SPACE      = 0;                                   SIM04690
              IF STEP.CORE_SIZE = 60 THEN STEP.CCRE_SIZE = MIN_CORE;     SIM04700
              END;                                                       SIM04710
              GOTO LOOP;                                                 SIM04720
                                                                         SIM04730
                                                                         SIM04740
LOCP:   END;                                                             SIM04750
                                                                         SIM04760
        END GENERATE_JOBS;                                               SIM04770
                                                                         SIM04780
                                                                         SIM04790
                                                                         SIM04800
(NOFIXEDOVERFLOW):                                                       SIM04809
RANDU:  PROCEDURE(I1,I2,X);                                              SIM04810
        /* RANDOM NUMBER GENERATOR */                                    SIM04820
                                                                         SIM04830
        DCL   I1    FIXED BIN(31),                                       SIM04840
              I2    FIXED BIN(31),                                       SIM04850
              X;                                                         SIM04860
                                                                         SIM04870
        I2 = I1 * 65539;                                                 SIM04880
        IF I2 < 0 THEN I2 = I2 + 2147483647 + 1;                         SIM04890
        X = I2;                                                          SIM04900
        X = X * .4656613E-9;                                             SIM04910
                                                                         SIM04920
        END RANDU;                                                       SIM04930
```

```
GET_MAIN:    PROCEDURE(I, CORESIZE);
        /* ALLOCATE MAIN STORAGE */

        DCL   CORESIZE FIXED BIN,
              N        FIXED BIN,
              I        FIXED BIN;

        CORE_HIGH(I) = 0;
        CORE_LOW(I)  = 0;
        N            = 1;
        DO WHILE (FREE_CORE(N) ¬= 0);
            IF FREE_CORE(N) - FREE_CORE(N+1) >= CORESIZE
               THEN
               DO;
               CORE_HIGH(I) = FREE_CORE(N);
               FREE_CORE(N) = FREE_CORE(N) - CORESIZE;
               CORE_LOW(I)  = FREE_CORE(N);
               HIGH(17)     = CORE_HIGH(I);
               LOW(17)      = CORE_LOW(I);
               INIT(17)     = I;
               N            = 1;
               DO WHILE (HIGH(N) ¬= 0);
                   IF HIGH(N) < HIGH(17)
                      THEN
                      DO;
                      USED_CORE(18) = USEC_CORE(N);
                      USED_CORE(N)  = USEC_CORE(17);
                      USED_CORE(17) = USEC_CORE(18);
                      END;
                   N = N + 1;
               END;
               USED_CORE(N) = USED_CORE(17);
               N            = 31;
               END;
            N = N + 2;
        END;

        END GET_MAIN;
```

```
FREE_MAIN:   PROCEDURE(I);
        /* FREE MAIN STORAGE */

        DCL   COMP FIXED BIN,
              SAVE FIXED BIN,
              N    FIXED BIN,
              I    FIXED BIN;

        DO COMP = CORE_HIGH(I), CORE_LCW(I);
              N = 1;
              DO WHILE (FREE_CORE(N) ¬= 0);
                    IF FREE_CORE(N) = COMP
                          THEN FREE_CORE(N), COMP = FREE_CORE(N+1);
                    IF FREE_CORE(N) < CCMP
                          THEN
                          DO;
                          SAVE      = FREE_CORE(N);
                          FREE_CORE(N) = COMP;
                          COMP      = SAVE;
                          END;
                    N = N + 1;
              END;
              FREE_CORE(N) = COMP;
        END;
        COMP = CORE_HIGH(I);
        N    = 1;
        DO WHILE (HIGH(N) ¬= 0);
              IF HIGH(N) = COMP
                    THEN
                    DO;
                    USED_CCRE(N) = USEC_CORE(N+1);
                    COMP = HIGH(N+1);
                    END;
              N = N + 1;
        END;

        END FREE_MAIN;
```

```
INIT_QUEUE:  PROCEDURE;
        /* INITIALIZE THE JOB INPUT QUEUE */

        DCL  I    FIXED BIN;

        DO I=1 TO JOB_QUEUE.SIZE;
             JOB_LINK(I) = I + 1;
        END;
        JOB_LINK(JOB_QUEUE.SIZE) = 0;
        FREE_HEAD = 1;
        AMOUNT    = 0;
        HEAD      = 0;

        END INIT_QUEUE;


ALLCC:    PROCEDURE ;
        /* ALLOCATE SPACE IN THE JOB INPUT QUEUE */

        DCL  I    FIXED BIN;

        I = FREE_HEAD;
        IF I = 0
             THEN /* SIMULATION QUEUE FULL */
             DO;
             PUT SKIP LIST ('SIMULATICN QUEUE FULL. RESTART.');
             PUT SKIP;
             GOTO SUPERVISOR;
             END;
        FREE_HEAD = JOB_LINK(I);
        RETURN(I);

        END ALLCC;
```

```
ENQUEUE:      PROCEDURE;
        /* ENQUEUE A NEW JOB INTO THE JOB QUEUE */
        /* ACCORDING TO IT'S CLASS AND PRIORITY */

        DCL   NEXT_IN     FIXED BIN,
              NEXT        FIXED BIN,
              LAST        FIXED BIN,
              IN          FIXED BIN,
              NEW         FIXED BIN;

        /* FIND POSITION IN QUEUE */

        AMOUNT(JOB.CLASS) = AMOUNT(JOB.CLASS) + 1;
        IN    = JOBS(JOB_COUNT);
        NEXT = HEAD(JOB.CLASS);
        LAST = 0;
        DO WHILE (NEXT ¬= 0);
              IF JOB.PRIORITY <= QUEUE.PRIORITY(NEXT)
                    THEN
                    DO;
                    LAST = NEXT;
                    NEXT = JOB_LINK(NEXT);
                    END;
              ELSE NEXT = 0;
        END;

        /*ENQUEUE JOB DESCRIPTION INTO GIVEN POSITION */

        NEW = ALLOC;
        IF LAST = 0
              THEN        /* JOB GOES ON TOP OF QUEUE */
              DO;
              JOB_LINK(NEW)     = HEAD(JOB.CLASS);
              HEAD(JOB.CLASS) = NEW;
              END;
        ELSE        /* JOB GOES ELSEWHERE IN THE QUEUE */
              DO;
              JOB_LINK(NEW)   = JOB_LINK(LAST);
              JOB_LINK(LAST) = NEW;
              END;
        QUEUE(NEW) = INPUT(IN);
        QUEUE.ARRIVAL(NEW) = TIME.SYSTEM;
```

```
        /* ENQUFUE ALL STEP DESCRIPTIONS */

        DO I=1 TO JOB.STEPS;
              STEP_LINK(NEW)  = ALLOC;
              NEW             = STEP_LINK(NEW);
              QUEUE(NEW)      = INPUT(IN + I);
        END;
        STEP_LINK(NEW) = 0;

        END ENQUEUE;


DEQUEUE:      PROCEDURE(I);
        /* DEQUEUE A JOB */

        DCL   N     FIXED BIN,
              M     FIXED BIN,
              I     FIXED BIN;

        JOB_INDEX(I)  = 0;
        DO N=1 TO 8;
              M = IN(I).CLASS(N);
              IF M ¬= 0
                    THEN
                    DO;
                    IF HEAD(M) ¬= 0
                          THEN        /* JOB FOUND */
                          DO;
                          JOB_INDEX(I)    = HEAD(M);
                          STEP_INDEX(I)   = STEP_LINK(HEAD(M));
                          JOB_NO(I)       = QUEUE.NUMBER(JOB_INDEX(I));
                          HEAD(M)         = JOB_LINK(HEAD(M));
                          AMOUNT(M)       = AMOUNT(M) - 1;
                          N               = 8;
                          END;
              END;
        END;

        END DEQUEUE;
```

```
FREE_QUEUE:  PROCEDURE(I);
        /* ADD QUEUE SPACE TO THE FREE LIST */

        JOB_LINK(JOB_INDEX(I)) = FREE_HEAD;
        FREE_HEAD = JOB_INDEX(I);
        DO WHILE (STEP_LINK(FREE_HEAD) ¬= 0);
             JOB_LINK(STEP_LINK(FREE_HEAD)) = FREE_HEAD;
             FREE_HEAD = STEP_LINK(FREE_HEAD);
        END;

        END FREE_QUEUE;



(NOFIXEDOVERFLOW):
CHANGE_SEED:  PROCEDURE;
        /* CHANGE SEED OF RANDOM NUMBER GENERATOR */

        DCL  1 CLOCK,
             2 DATE     CHAR(8),
             2 TIME     CHAR(8),
             2 V_TIME   FIXED BIN(31),
             2 T_TIME   FIXED BIN(31);

        CALL IXECLOK(CLOCK);
        V_TIME = V_TIME * 10;
        SEED_1 = V_TIME + SEED_1;
        SEED_2 = V_TIME + SEED_2;

        END CHANGE_SEED;
```

```
START: PROCEDURF;                                                       SIM07030
        /* INITIALIZE FOR A NEW RUN */                                  SIM07040
                                                                        SIM07050
        DCL   1 MOD,                                                    SIM07051
                2 SPOOL    FIXED BIN(31),                               SIM07052
                2 SPACE    FIXED BIN(31),                               SIM07053
                2 CORE_H   FIXED BIN,                                   SIM07054
                2 CORE_L   FIXED BIN,                                   SIM07055
                2 DISKS    FIXED BIN,                                   SIM07056
                2 TAPES    FIXED BIN;                                   SIM07057
                                                                        SIM07058
        DCL   N            FIXED BIN;                                   SIM07060
                                                                        SIM07070
        TIME      = 0;                                                  SIM07080
        TIME_LC   = 0;                                                  SIM07090
        IN_QUEUF  = 0;                                                  SIM07100
        N         = 0;                                                  SIM07110
        SYSTEM.FREE_CORE = 0;                                           SIM07120
        PUT SKIP LIST ('SYSTEM MODIFICATIONS? (1=YES, 0=NO)');          SIM07130
        PUT SKIP;                                                       SIM07140
        GET LIST (N);                                                   SIM07150
        IF N=1                                                          SIM07160
            THEN                                                        SIM07170
            DO;                                                         SIM07180
            MOD = INITIAL_VALUES;                                       SIM07190
            PUT SKIP LIST ('ENTER:');                                   SIM07200
            PUT SKIP;                                                   SIM07210
            GET DATA;                                                   SIM07220
            INITIAL_VALUES = MOD;                                       SIM07230
            END;                                                        SIM07240
        SYSTEM.TAPES          = I_TAPES;                                SIM07250
        SYSTEM.D_A_SPACE      = I_D_A_SPACE;                            SIM07260
        SYSTEM.USED_CORE      = 0;                                      SIM07270
        SYSTEM.FREE_CORE(1) = I_CORE_HIGH;                              SIM07280
        SYSTEM.FREE_CORE(2) = I_CORE_LOW;                               SIM07290
        SYSTEM.DISKS          = I_DISKS;                                SIM07300
        SYSTEM.IN_SPOOL       = I_IN_SPOOL;                             SIM07310
        STATUS    = 2;                                                  SIM07320
        RESOURCES = 0;                                                  SIM07330
        DO I=1 TO 15;                                                   SIM07340
            IF STOP_SWITCH(I) >= 0 THEN STOP_SWITCH(I) = 0;             SIM07350
            ELSE STOP_SWITCH(I) = -2;                                   SIM07360
            END;                                                        SIM07370
        CALL INIT_QUEUE;                                                SIM07380
                                                                        SIM07390
        END START;                                                      SIM07400
```

```
SET_SIM_PARAMETERS:    PROCEDURE;
        /* ALLOW RUN PARAMETERS TO BE ENTERED */

        PUT SKIP LIST ('RUN PARAMETERS:');
        PUT SKIP LIST ('TIME (0 < SEC. < 604801) :');
        PUT SKIP;
        GET LIST (DELTA_TIME);
        DO WHILE (DELTA_TIME < 1 | DELTA_TIME > 604800);
             PUT LIST ('*  INVALID INPUT. RETRY:');
             PUT SKIP;
             GET LIST (DELTA_TIME);
        END;

        PUT SKIP LIST ('JOBS (0 < NUMBER < 1001) :');
        PUT SKIP;
        GET LIST (MAX_JOBS);
        DO WHILE (MAX_JOBS < 1 | MAX_JOBS > 1000);
             PUT LIST ('*  INVALID INPUT. RETRY:');
             PUT SKIP;
             GET LIST (MAX_JOBS);
        END;

        END SET_SIM_PARAMETERS;



SET_RESTART:  PROCEDURE;

        PUT SKIP;
        PUT LIST ('RESTART PARAM. (0 - 6) : ') SKIP;
        PUT SKIP;
        GET LIST (RESTART);
        DO WHILE (RESTART < 0 | RESTART > 6);
             PUT LIST ('* INVALID INPUT. RETRY: ') SKIP;
             PUT SKIP;
             GET LIST (RESTART);
        END;

        END SET_RESTART;
```

```
INITIALIZE: PROCEDURE;
      /* INITIALIZE THE SIMULATION MODEL */

INITIALIZE_1:     ENTRY;

        OPEN FILE (P1) PAGESIZE(75);
        SEED_1             = 98765;
        SEED_2             = 12345;
        ALPHA              = 1.3083 ;
        STATISTICS.MAXI    = 0;
        STATISTICS.NO      = 0;
        STOP_SWITCH        = -2;
        IN.CLASS           = 0;
        CLASSES            = '  STCP  ';
        RETURN;

INITIALIZE_2:     ENTRY;

        IF RESTART = 3 | RESTART = 6     /* CHANGE SEED FOR JOB GEN.*/
           THEN CALL CHANGE_SEED;
        IF RESTART < 4                   /* EMPTY QUEUE FCR NEW RUN */
           THEN CALL START;
        IF RESTART ¬= 2 & RESTART ¬= 5   /*    GET SIM. PARAMETERS   */
           THEN                          /* GENERATE NEW JOB STREAM */
           DO;
           CALL SET_SIM_PARAMETERS;
           CALL GENERATE_JOBS;
           END;
        TIME.READER = TIME.SYSTEM + INPUT.DEL_ARRIVAL(1);
        JCB_CCUNT   = 1;
        JOB_PTR     = ADDR(INPUT);
        TIME.STOP   = TIME.SYSTEM + DELTA_TIME;
        RETURN;

        END INITIALIZE;
```

```
MODIFY:     PROCEDURE;
        /* ALLOW INITIATOR MODIFICATIONS TO BE ENTERED */

        DCL   N             FIXED BIN,
              CHECK         FIXED BIN,
              TEMP          FIXED BIN,
              I             FIXED BIN;

        PUT SKIP LIST ('MODIFY INITIATORS:');
LOOP:
        PUT SKIP LIST ('ENTER N (1-15 = INIT.# , 0 = END MOD.) :');
        PUT SKIP;
        GET LIST (N);
        IF N < 1 | N > 15 THEN GOTO SET_TRACE;

        PUT SKIP LIST ('ENTER JOB CLASSES ''A-O''  OR ''STOP'' :');
        PUT SKIP;
        GET LIST (CLASSES(N));
        CHECK = VERIFY(CLASSES(N),'ABCDEFGHIJKLMNO ');
        IF CHECK ¬= 0
              THEN /* STOP INITIATOR */
              DO;
              CLASSES(N) = '  STOP  ';
              IF STOP_SWITCH(N) >= 0
                    THEN STOP_SWITCH(N) = STOP_SWITCH(N) - 2;
              END;
        ELSE /* SET NEW JOB CLASSES */
              DO;
              IF STOP_SWITCH(N) = -2
                    THEN TIME_LC(N) = TIME.SYSTEM;
              IF STOP_SWITCH(N) = 0 | STOP_SWITCH(N) = -2
                    THEN TIME.INITIATOR(N) = TIME.SYSTEM;
              IF STOP_SWITCH(N) < 0
                    THEN STOP_SWITCH(N) =  STOP_SWITCH(N) + 2;
              DO I=1 TO 8;
                    TEMP = UNSPEC(SUBSTR(CLASSES(N),I,1));
                    IF TEMP < 202
                          THEN TEMP = TEMP - 192;
                    ELSE TEMP = TEMP - 199;
                    IF TEMP < 0
                          THEN TEMP = 0;
                    IN(N).CLASS(I) = TEMP;
              END;
        END;
        GOTO LOOP;
```

```
SET_TRACE:
        PUT SKIP LIST ('SET TRACE, CORE MAP, STAT. PARAM. (1=ON) :');
        PUT SKIP;
        GET LIST (TRACE, CORE_MAP,  RESULTS);
        IF TRACE = 1
            THEN PUT FILE (P1) PAGE LINE(9) EDIT ('*    ') (X(15),A(3));

        END MODIFY;



 CP_RESPONSE:     PROCEDURE(I);
        /* SIMULATE OPERATOR RESPONSE */

        DCL DISK_MOUNT(31) INIT( .0000, .0197, .0724, .1513, .2829,
                                 .3487, .4868, .5724, .6250, .6842,
                                 .7434, .7566, .7961, .8092, .8355,
                                 .8487, .8618, .8816, .9079, .9276,
                                 .9342, .9403, .9474, .9539, .9605,
                                 .9671, .9737, .9803, .9868, .9934,
                                 1.00);

        DCL TAPE_MOUNT(31) INIT( .0000, .0733, .2454, .4249, .5421,
                                 .6264, .6996, .7253, .7509, .7839,
                                 .8168, .8352, .8681, .8828, .8938,
                                 .9121, .9267, .9377, .9451, .9524,
                                 .9497, .9670, .9707, .9744, .9780,
                                 .9817, .9853, .9890, .9927, .9963,
                                 1.00);

        DCL ANSWER(11) INIT(     .6715, .6968, .7401, .7978, .8159,
                                 .8267, .8484, .8626, .8700, .8917,
                                 1.00);

        DCL CANCEL(2) INIT( .0130, 1.00);

        DCL X;
```

```
REAL2:  PROCEDURE(TABLE);

        DCL  TABLE(60), NUMBER, VALUE, RANDOM_X, X_LOW;

        CALL RANDU(SEED_2,SEED_2,RANDOM_X);
        NUMBER = 1;
        DO WHILE (RANDOM_X > TABLE(NUMBER));
             NUMBER = NUMBER + 1;
        END;
        IF NUMBER = 1 THEN X_LOW = 0;
        ELSE X_LOW = TABLE(NUMBER-1);
        VALUE = NUMBER - 1 + (RANDOM_X - X_LOW)
             / (TABLE(NUMBER) - X_LOW);
        RETURN(VALUE);

        END REAL2;



        OP_TIME(I) = 0;
        OP_ANSWER(I) = REAL2(CANCEL);
        IF STEP.DISKS > I_DISKS | STEP.TAPES > I_TAPES
             THEN OP_ANSWER(I) = 0;
        IF OP_ANSWER(I) = 0 THEN OP_ANSWER(I) = -1;
        IF STEP.OP_REQUESTS ¬= 0 | OP_ANSWER(I) < 0
             THEN
             DO;
             X = REAL2(ANSWER);
             IF X < 10 THEN OP_TIME(I) = X * 20;
             ELSE OP_TIME(I) = (X - 10) * 620 + 200;
             END;
        IF OP_ANSWER(I) < 0 | STATUS(I) = 4 THEN RETURN;
        IF STEP.TAPES ¬= 0 THEN OP_TIME(I) = REAL2(TAPE_MOUNT)
             * 20 * STEP.TAPES + .5 + OP_TIME(I);
        IF STEP.DISKS ¬= 0 THEN OP_TIME(I) = REAL2(DISK_MOUNT)
             * 20 * STEP.DISKS + .5 + OP_TIME(I);
        IF OP_TIME(I) > 1000 THEN OP_TIME(I) = 1000;

        END OP_RESPONSE;
```

```
WRITE_STATISTICS:     PROCEDURE;
          /* WRITE STATISTICS */

          DCL  SAVE_1     FIXED BIN,
               SAVE_2     FIXED BIN,
               I          FIXED BIN,
               N          FIXED BIN;

          TIME.SYSTEM = MIN(TIME.SYSTEM, TIME.STOP);
          IF RESULTS  = 1
               THEN       /* GATHER STATISTICAL RESULTS */
               DO;
               TIME_SNAP    = TIME.SYSTEM;
               SAVE_1       = TRACE;
               DO I=1 TO 15;
                    SAVE_2 = OP_ANSWER(I);
                    IF STOP_SWITCH(I) = -2 THEN GOTO NEXT_I;
                    IF STATUS(I) = 1 THEN CALL STATISTIC_END_WW(I);
                    IF STATUS(I) = 2 | STATUS(I) = 3
                                     THEN CALL STATISTIC_END_WM(I);
                    IF STATUS(I) = 4 THEN CALL STATISTIC_END_WC(I);
                    IF STATUS(I) = 5 THEN CALL STATISTIC_END_WV(I);
                    IF STATUS(I) = 6 THEN CALL STATISTIC_END_WS(I);
                    IF STATUS(I) = 7 THEN CALL STATISTIC_END_STEP(I);
                    OP_ANSWER(I) = SAVE_2;
                    TIME_LC(I)   = TIME.SYSTEM;
NEXT_I:        END;
               WRITE FILE (STAT) FROM (STATISTICS);
               STATISTICS.MAXI = 0;
               STATISTICS.NO   = 0;
               IN_QUEUE        = AMOUNT;
               TRACE           = SAVE_1;
               END;
```

```
IF CORE_MAP ¬= 1 THEN RETURN;
/* PRINT OUT A CORE MAP */

PUT FILE (P1) SKIP(6) EDIT (' USAGE OF MAIN MEMORY')
(X(15),A(21));
PUT FILE (P1) SKIP(2) EDIT (' HIGH  LOW CORE  INIT.# JOB #')
(X(15),A(29));
PUT FILE (P1) SKIP(2);
IF HIGH(1) = 0
     THEN
     DO;
     CORE = I_CORE_HIGH - I_CORE_LOW;
     PUT FILE (P1) SKIP EDIT
     (I_CORE_HIGH,I_CORE_LOW,CORE,' FREE')
     (X(15), 3 F(5), X(3), A(5));
     RETURN;
     END;
CORE = I_CORE_HIGH - HIGH(1);
IF HIGH(1) < I_CORE_HIGH THEN PUT FILE (P1) SKIP EDIT
     (I_CORE_HIGH,HIGH(1),CORE,'FREE')(X(15),3 F(5),X(4),A(4));
N = 1;
DO WHILE (HIGH(N) ¬= 0);
     CORE = HIGH(N) - LOW(N);
     PUT FILE (P1) SKIP EDIT (HIGH(N),LOW(N),CORE,INIT(N),
     JOB_NO(INIT(N))) (X(15),3 F(5),X(3),2 F(5));
     IF LOW(N) ¬= HIGH(N+1) & HIGH(N+1) ¬= 0
          THEN
          DO;
          CORE = LOW(N) - HIGH(N+1);
          PUT FILE (P1) SKIP EDIT
          (LOW(N),HIGH(N+1),CORE,' FREE')
          (X(15),3 F(5),X(3),A(5));
          END;
     N = N + 1;
END;
IF LOW(N-1) > I_CORE_LOW
     THEN
     DO;
     CORE = LOW(N-1) - I_CORE_LOW;
     PUT FILE (P1) SKIP EDIT
     (LOW(N-1),I_CORE_LOW,CORE,'FREE')(X(15),3 F(5),X(4),A(4));
     END;
RETURN;

END WRITE_STATISTICS;
```

```
STATISTIC:   PRCCEDURE;
        /* COLLECT STATISTICAL DATA */

        DCL   WAIT FIXED BIN,
              I    FIXED BIN;


STATISTIC_START_JOB:    ENTRY(I);
        /* JOB STARTED */

        JOB_START(I,JOB.CLASS) = JOB_START(I,JOB.CLASS) + 1;
        WAIT                   = TIME.SYSTEM - JOB.ARRIVAL;
        JOB_WAIT(JOB.CLASS)    = JOB_WAIT(JOB.CLASS) + WAIT;
        JOB_WAITS(JOB.CLASS)   = JOB_WAITS(JCB.CLASS) + WAIT ** 2;
        JOB_WAITM(JOB.CLASS)   = MAX (JOB_WAITM(JOB.CLASS), WAIT);
        IF TRACE = 1
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' STARTED BY INITIATOR ',
             I,'*  ') (F(7),A(6),F(6),A(22),F(2),SKIP,X(15),A(3));
        RETURN;


STATISTIC_END_STEP:     ENTRY(I);
        /* UPDATE TOTAL TIME SERVING JOBS */

        WAIT = TIME.SYSTEM - TIME_LC(I);
        MAX_SJ(I) = MAX (MAX_SJ(I), WAIT);
        TIME_SJ(I) = TIME_SJ(I) + WAIT;
        RETURN;


STATISTIC_END_JOB:      ENTRY(I);
        /* JOB TERMINATED */

        IF TRACE = 1
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' TERMINATED','*   ')
             (F(7),A(6),F(6),A(11),SKIP,X(15),A(3));
        RETURN;
```

```
STATISTIC_BEGIN_WW:     ENTRY(I);
        /* NO JOBS TO SERVE. WAITING FOR WORK */

        IF TRACE = 1
            THEN PUT FILE (P1) EDIT
            (TIME.SYSTEM,' * INIT.   ',I,' WAITING FOR WORK','*   ')
            (F(7),A(10),F(2),A(17),SKIP,X(15),A(3));
        RETURN;


STATISTIC_END_WW: ENTRY(I);
        /* UPDATE TOTAL TIME WAITING FOR WORK */

        WAIT = TIME.SYSTEM - TIME_LC(I);
        MAX_WW(I) = MAX (MAX_WW(I), WAIT);
        TIME_WW(I) = TIME_WW(I) + WAIT;
        RETURN;


STATISTIC_END_WM: ENTRY(I);
        /* UPDATE TOTAL TIME WAITING FOR MAIN MEMORY */

        WAIT = TIME.SYSTEM - TIME_LC(I);
        MAX_WM(I) = MAX (MAX_WM(I), WAIT);
        TIME_WM(I) = TIME_WM(I) + WAIT;
        RETURN;
```

```
STATISTIC_BEGIN_WD:     ENTRY(I);
        /* NOT ALL DEVICES AVAILABLE. SIMULATE */
        /* OPERATOR RESPONSE TIME AND ANSWER    */

        CALL OP_RESPONSE(I);
        IF TRACE = 1 & STEP.DISKS ¬= 0
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' WAITING FOR ',
             STEP.DISKS,' DISK(S)   ','*  ')
             (F(7),A(6),F(6),A(13),F(2),A(10),SKIP,X(15),A(3));
        IF TRACE = 1 & STEP.TAPES ¬= 0
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' WAITING FOR ',
             STEP.TAPES,' TAPE(S)   ','*  ')
             (F(7),A(6),F(6),A(13),F(2),A(10),SKIP,X(15),A(3));
        RETURN;



STATISTIC_END_WD: ENTRY(I);
        /* UPDATE TOTAL TIME WAITING FOR DEVICES */

        WAIT = TIME.SYSTEM - TIME_LC(I);
        MAX_WD(I) = MAX (MAX_WD(I), WAIT);
        TIME_WD(I) = TIME_WD(I) + WAIT;
        OP_ANSWER(I) = 0;
        RETURN;



STATISTIC_ABEND_DEVICE:     ENTRY(I);
        /*JOB CANCELLED BY OPERATOR BECAUSE OF LACK OF DEVICES */

        ABEND_DEVICE(I) = ABEND_DEVICE(I) + 1;
        IF TRACE = 1
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' CANCELLED BY OPERTOR',
             '*  ') (F(7),A(6),F(6),A(22),SKIP,X(15),A(3));
        RETURN;
```

```
STATISTIC_BEGIN_WV:     ENTRY(I);
        /* ALLOCATE DATA SETS.  SIMULATE OPERATCR RESPCNSE TIME`*/

        CALL OP_RESPONSE(I);
        IF TRACE = 1 & STEP.TAPES ¬= 0
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JCB.NUMBER,' : MOUNT ',
             STEP.TAPES,' TAPE(S)','*  ')
             (F(7),A(6),F(6),A(9),F(2),A(9),SKIP,X(15),A(3));
        IF TRACE = 1 & STEP.DISKS ¬= 0
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JCB.NUMBER,' : MOUNT ',
             STEP.DISKS,' DISK(S)','*  ')
             (F(7),A(6),F(6),A(9),F(2),A(9),SKIP,X(15),A(3));
        IF TRACE = 1 & STEP.OP_REQUESTS ¬= 0
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JCB.NUMBER,' : ALLOCATE 1 ',
             'DATA SET','*  ')
             (F(7),A(6),F(6),A(14),A(8),SKIP,X(15),A(3));
        RETURN;


STATISTIC_END_WV: ENTRY(I);
        /* UPDATE TOTAL TIME WAITING FOR DATA SET ALLOCATION */

        OP_ANSWER(I) = 0;
        WAIT = TIME.SYSTEM - TIME_LC(I);
        MAX_WV(I) = MAX (MAX_WV(I), WAIT);
        TIME_WV(I) = TIME_WV(I) + WAIT;
        IF TRACE = 1 & ( IN.TAPES(I) ¬= 0 | IN.CISKS(I) ¬= 0 )
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' ALL VOLUMES MOUNTED',
             '*  ') (F(7),A(6),F(6),A(20),SKIP,X(15),A(3));
        IF TRACE = 1 & STEP.OP_REQUESTS ¬= 0
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' DATA SETS ALLOCATED',
             '*  ') (F(7),A(6),F(6),A(20),SKIP,X(15),A(3));
             RETURN;


 STATISTIC_ABEND_DATA_SET:  ENTRY(I);
        /* JOB CANCELLED BY OPERATOR, CATA SET(S) NOT AVAILABLE */

        ABEND_D_SET(I) = ABEND_D_SET(I) + 1;
        IF TRACE = 1
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' CANCELLED BY OPERATOR',
             '*  ') (F(7),A(6),F(6),A(22),SKIP,X(15),A(31));
        RETURN;
```

```
STATISTIC_END_WS: ENTRY(I);
        /* UPDATE TOTAL TIME WAITING FOR D.A.SPACE */

        WAIT = TIME.SYSTEM -TIME_LC(I);
        MAX_WS(I) = MAX (MAX_WS(I), WAIT);
        TIME_WS(I) = TIME_WS(I) + WAIT;
        RETURN;



STATISTIC_ABEND_D_A_SPACE:  ENTRY(I);
        /* JOB ABENDED BECAUSE OF LACK OF D.A.SPACE */

        ABEND_SPACE(I) = ABEND_SPACE(I) + 1;
        IF TRACE = 1
             THEN PUT FILE (P1) EDIT
             (TIME.SYSTEM,' * JOB',JOB.NUMBER,' ABEND: NO SPACE',
             '*  ') (F(7),A(6),F(6),A(20),SKIP,X(15),A(3));
        RETURN;

        END STATISTIC;
```

```
READER:     PROCEDURE;
      /* SIMULATE THE READER/INTERPRETER FUNCTIONS */

      JOB_PTR = ADDR(INPUT(JOBS(JOB_COUNT)));
      IF IN_SPOOL >= JOB.CARDS
           THEN    /* ENOUGH INPUT SPOOL SPACE, READ JCB */
           DO;
           JOB_IN(JOB.CLASS) = JOB_IN(JOB.CLASS) + 1;
           JOB.ARRIVAL       = TIME.SYSTEM;
           IN_SPOOL          = IN_SPOOL - JCB.CARDS;
           CALL ENQUEUE;
           JOB_COUNT = JCB_COUNT + 1;
           IF JOB_COUNT <= MAX_JOBS
                THEN     /* MORE JOBS TO READ */
                DO;
                JOB_PTR     = ADDR(INPUT(JOBS(JOB_COUNT)));
                TIME.READER = JOB.CEL_ARRIVAL + TIME.READER;
                END;
           ELSE;    /* ALL JOBS READ IN */
           END;
      ELSE /* NOT ENOUGH INPUT SPOCL SPACE    */
           /* WAIT UNTIL NEXT JOB TERMINATES */
           TIME.READER = NEXT_EOJ;

      END READER;
```

```
INITIATOR:  PROCEDURE(I);

        DCL  ACTIVE      FIXED BIN,
             I           FIXED BIN,
             N           FIXED BIN,
             STATE(8)    LABEL INIT
                         (WAIT_FOR_WORK,          JOB_SELECTION,
                         REGION_MANAGEMENT,       DEVICE_ALLCCATICN,
                         DATA_SET_ALLOCATION,     D_A_SPACE_ALLOCATION,
                         STEP_TERMINATICN,        JCB_TERMINATION);


        JOB_PTR  = ADDR( QUEUE(JOB_INCEX(I)));;
        STEP_PTR = ADDR( QUEUE(STEP_INDEX(I)));;
        GOTC STATE(STATUS(I));


WAIT_FCR_WORK:

        CALL GET_MAIN(I,MIN_CORE);
        IF CORE_HIGH(I) = 0
             THEN /* NOT ENOUGH CORE */
             DO;
             TIME.INITIATOR(I) = NEXT_EOJ;
             RETURN;
             END;
        CALL DEQUEUE(I);
        IF JOB_INDEX(I) = 0
             THEN /* NO JOBS TO SERVE */
             DO;
             TIME.INITIATOR(I) = TIME.READER;
             CALL FREE_MAIN(I);
             RETURN;
             END;

        /* JOB FOUND */

        JOB_PTR = ADDR(QUEUE(JOB_INCEX(I)));
        CALL STATISTIC_START_JOB(I);
        CALL STATISTIC_END_WW(I);
        STOP_SWITCH(I) = 1;
        TIME_LC(I)     = TIME.SYSTEM;
        STATUS(I)      = 3;
        RETURN;
```

```
JCB_SELECTICN:

        CALL GET_MAIN(I,MIN_CORE);
        IF CORE_HIGH(I) = 0
             THEN /* NOT ENOUGH CORE */
             DO;
             TIME.INITIATOR(I) = NEXT_EOJ;
             RETURN;
             END;
        CALL STATISTIC_END_WM(I);
        TIME_LC(I) = TIME.SYSTEM;
        CALL DEQUEUE(I);
        IF JOB_INDEX(I) = 0
             THEN /* NO JOBS TO SERVE */
             DO;
             CALL STATISTIC_BEGIN_WW(I);
             TIME.INITIATOR(I) = TIME.READER;
             STATUS(I)         = 1;
             STOP_SWITCH(I)    = 0;
             CALL FREE_MAIN(I);
             RETURN;
             END;

        /* JOB FOUND */

        JOB_PTR  = ADDR(QUEUE(JOB_INDEX(I)));
        STEP_PTR = ADDR(QUEUE(STEP_INDEX(I)));
        CALL STATISTIC_START_JOB(I);
        STOP_SWITCH(I) = 1;
        STATUS(I)      = 3;
```

```
REGION_MANAGEMENT:
        /* FREE OLD REGION, GET NEW REGION */

        CALL FREE_MAIN(I);
        CALL GET_MAIN(I,STEP.CORE_SIZE);
        IF CORE_HIGH(I) = 0
            THEN /* CORE NOT AVAILABLE */
            DO;
            TIME.INITIATOR(I) = NEXT_EOJ;
            RETURN;
            END;

        /* NEW REGION ALLOCATED */

        CALL STATISTIC_END_WM(I);
        TIME_LC(I) = TIME.SYSTEM;
        STATUS(I)  = 4;



DEVICE_ALLOCATION:
        /* ALLOCATE REQUESTED DIVICES */

        IF CP_ANSWER(I) < 0
            THEN /* JOB CANCELLED BY OPERATOR */
            DO;
            CALL STATISTIC_END_WD(I);
            CALL STATISTIC_ABEND_DEVICE(I);
            TIME_LC(I) = TIME.SYSTEM;
            STATUS(I)  = 8;
            RETURN;
            END;
        SYSTEM.DISKS = SYSTEM.DISKS - STEP.DISKS;
        IN.DISKS(I)  = IN.DISKS(I) +  STEP.DISKS;
        SYSTEM.TAPES = SYSTEM.TAPES - STEP.TAPES;
        IN.TAPES(I)  = IN.TAPES(I)  + STEP.TAPES;
        STEP.DISKS   = 0;
        STEP.TAPES   = 0;
```

```
IF SYSTEM.TAPES >= 0 & SYSTEM.DISKS >= C
   THEN /* ALL DEVICES AVAILABLE */
   DO;
   CALL STATISTIC_END_WD(I);
   STEP.DISKS = IN.DISKS(I);
   STEP.TAPES = IN.TAPES(I);
   TIME_LC(I) = TIME.SYSTEM;
   STATUS(I)  = 5;
   GOTO DATA_SET_ALLOCATICN;
   END;

/* NOT ALL DEVICES AVAILABLE */
IF SYSTEM.DISKS < 0
   THEN
   DO;
   STEP.DISKS   = -SYSTEM.DISKS;
   IN.DISKS(I)  = IN.DISKS(I) - STEP.CISKS;
   SYSTEM.DISKS = 0;
   END;
IF SYSTEM.TAPES < 0
   THEN
   DC;
   STEP.TAPES  = -SYSTEM.TAPES;
   IN.TAPES(I) = IN.TAPES(I) - STEP.TAPES;
   SYSTEM.TAPES= 0;
   END;
IF OP_ANSWER(I) = 0
   THEN /* GET OPERATCR RESPONSE */
   DO;
   CALL STATISTIC_BEGIN_WC(I);
   TIME.INITIATOR(I) = TIME.INITIATCR(I) + OP_TIME(I);
   END;
ELSE TIME.INITIATOR(I) = NEXT_EOJ;
RETURN;
```

```
DATA_SET_ALLOCATION:
        7* ALLOCATE REQUESTED DATA SETS */

        IF OP_ANSWER(I) < 0
           THEN             /* JOB CANCELLED BY OPERATOR */
           DO;
           CALL STATISTIC_END_WV(I);
           CALL STATISTIC_ABEND_DATA_SET(I);
           TIME_LC(I) = TIME.SYSTEM;
           STATUS(I)  = 8;
           RETURN;
           END;

        /* ALLOCATE DATA SETS          */

        IF STEP.TAPES ¬= 0 | STEP.DISKS ¬= 0 | STEP.OP_REQUESTS ¬= 0
           THEN  /* SIMULATE OPERATOR RESPONSE TIME */
           DO;
           CALL STATISTIC_BEGIN_WV(I);
           TIME.INITIATOR(I) = TIME.INITIATOR(I) + OP_TIME(I);
           STEP.OP_REQUESTS  = 0;
           STEP.DISKS        = 0;
           STEP.TAPES        = 0;
           RETURN;
           END;

        /* ALL DATA SETS ALLOCATED                */

        CALL STATISTIC_END_WV(I);
        TIME_LC(I) = TIME.SYSTEM;
        STATUS(I)  = 6;
```

```
D_A_SPACE_ALLOCATION:
/* ALLOCATE SPACE ON DIRECT ACCESS DEVICES */

SYSTEM.D_A_SPACE= SYSTEM.D_A_SPACE - STEP.D_A_SPACE;
IN.D_A_SPACE(I)= IN.D_A_SPACE(I) + STEP.D_A_SPACE;
STEP.D_A_SPACE = 0;
IF SYSTEM.D_A_SPACE >= 0
     THEN /* D.A.SPACE AVAILABLE */
     DO;
     CALL STATISTIC_END_WS(I);
     TIME.INITIATOR(I)= TIME.INITIATOR(I) + STEP.RUN_TIME;
     TIME_LC(I)  = TIME.SYSTEM;
     STATUS(I)   = 7;
     RETURN;
     END;

/* NOT ENOUGH D.A.SPACE AVAILABLE */
/*  CHECK IF OTHER JOB IS ACTIVE  */

STEP.D_A_SPACE   = -SYSTEM.D_A_SPACE;
IN.D_A_SPACE(I)  = IN.D_A_SPACE(I) - STEP.D_A_SPACE;
SYSTEM.D_A_SPACE = 0;
ACTIVE = 0;
DO N=1 TO 15;
     IF STATUS(N) = 7 | STATUS(N) = 8 THEN ACTIVE = 1;
END;

IF ACTIVE ¬= 0
     THEN /* WAIT UNTIL NEXT JOB TERMINATES */
     DO;
     TIME.INITIATOR(I) = NEXT_EOJ;
     RETURN;
     END;

/* NO OTHER JOB ACTIVE. ABEND THIS JCB */

CALL STATISTIC_END_WS(I);
CALL STATISTIC_ABEND_D_A_SPACE(I);
TIME_LC(I) = TIME.SYSTEM;
STATUS(I)  = 8;
RETURN;
```

```
STEP_TERMINATION:
        /* TERMINATE STEPS */

        CALL STATISTIC_END_STEP(I);
        TIME_LC(I) = TIME.SYSTEM;
        STEP_INDEX(I) = STEP_LINK(STEP_INDEX(I));
        IF STEP_INDEX(I) = 0
             THEN /* NO MORE STEPS */
             DO;
             STATUS(I) = 8;
             RETURN;
             END;

        /* MORE STEPS */

        SYSTEM.DISKS = SYSTEM.DISKS + IN.DISKS(I);
        SYSTEM.TAPES = SYSTEM.TAPES + IN.TAPES(I);
        IN.TAPES(I)  = 0;
        IN.DISKS(I)  = 0;
        STATUS(I)    = 3;
        RETURN;
```

```
JCB_TERMINATICN:
      /* TERMINATE JOBS */

      SYSTEM.D_A_SPACE = SYSTEM.D_A_SPACE + IN.D_A_SPACE(I);
      SYSTEM.IN_SPOOL  = SYSTEM.IN_SPOOL + JOB.CARDS;
      SYSTEM.TAPES     = SYSTEM.TAPES + IN.TAPES(I);
      SYSTEM.DISKS     = SYSTEM.DISKS + IN.DISKS(I);
      CALL FREE_MAIN(I);
      CALL FREE_QUEUE(I);
      RESOURCES(I)     = 0;
      CALL STATISTIC_END_JOB(I);
      STATUS(I)        = 2;
      STOP_SWITCH(I)   = STOP_SWITCH(I) - 1;
      TIME_LC(I)       = TIME.SYSTEM;
      IF TIME.READER   = NEXT_EOJ THEN TIME.READER = TIME.SYSTEM;
      DO N=1 TO 15;
          IF TIME.INITIATOR(N) = NEXT_EOJ
              THEN TIME.INITIATOR(N) = TIME.SYSTEM;
      END;
      RETURN;



      END INITIATOR;
```

```
TIMER: PRCCEDURE;
        /* MAINTAIN THE TIME TABLE */

        DCL   I    FIXED BIN;

        IF TIME.SYSTEM = TIME.READER
            THEN CALL READER;
        DO I=1 TO 15;
            IF TIME.SYSTEM = TIME.INITIATOR(I) & STOP_SWITCH(I) > -2
                THEN
                DO;
                CALL  INITIATOR(I);
                END;
        END;

        /* UPDATE THE SYSTEM TIME */

        TIME.SYSTEM = TIME.READER;
        DO I=1 TO 15;
            IF TIME.SYSTEM > TIME.INITIATOR(I) & STOP_SWITCH(I) > -2
                THEN TIME.SYSTEM = TIME.INITIATCR(I);
        END;

        END TIMER;
```

```
SUPERVISOR:
        /* BODY OF SIMULATION PROGRAM */

        ON ENDFILE (SYSIN) BEGIN;
            CLOSE FILE (SYSIN);
            END;

        ON ENDPAGE (P1) BEGIN;
            PUT FILE (P1) PAGE;
            PUT FILE (P1) LINE(9);
            END;


        CALL INITIALIZE_1;
        DO WHILE (RESTART ¬= 0);
            CALL INITIALIZE_2;
            CALL MODIFY;
            DO WHILE (TIME.SYSTEM < TIME.STOP
                    & JOB_COUNT <= MAX_JOBS);
                CALL TIMER;
            END;
            IF TIME.SYSTEM =  NEXT_EOJ THEN TIME.SYSTEM = TIME.STOP;
            CALL WRITE_STATISTICS;
            CALL SET_RESTART;
        END;

        END SIM;
```

```
STA:     PROCEDURE OPTIONS(MAIN);
         /* EVALUATE AND PRINT STATISTICS */

         DCL  CLASS       CHAR(15) INIT ('ABCDEFGHIJKLMNO');

         DCL  SUM(7)      BIN FLOAT,
              IN(7)       BIN FIXED,
              T(7)        BIN FIXED;

         DCL  1 S,
              2 TIME_LC(15)   FIXED BIN(31),
              2 TIME_SNAP     FIXED BIN(31),
              2 MAXI(15),
              3 JOB_WAITM     FIXED BIN(31),
              3 MAX_SJ        FIXED BIN(31),
              3 MAX_WD        FIXED BIN(31),
              3 MAX_WM        FIXED BIN(31),
              3 MAX_WS        FIXED BIN(31),
              3 MAX_WV        FIXED BIN(31),
              3 MAX_WW        FIXED BIN(31),
              2 NO(15),
              3 TIME_SJ       FIXED BIN(31),
              3 TIME_WM       FIXED BIN(31),
              3 TIME_WD       FIXED BIN(31),
              3 TIME_WV       FIXED BIN(31),
              3 TIME_WS       FIXED BIN(31),
              3 TIME_WW       FIXED BIN(31),
              3 JOB_WAIT      FIXED BIN(31),
              3 JOB_WAITS     FLOAT BIN,
              3 JOB_START(15)FIXED BIN,
              3 IN_QUEUE      FIXED BIN,
              3 ABEND_DEVICE  FIXED BIN,
              3 ABEND_D_SET   FIXED BIN,
              3 ABEND_SPACE   FIXED BIN,
              3 OP_ANSWER     FIXED BIN,
              3 CP_TIME       FIXED BIN,
              3 JOB_IN        FIXED BIN,
              2 CLASSES(15)   CHAR  (8);

         DCL  1 TOTAL,
              2 CURRENT_SNAP  FIXED BIN(31),
              2 FIRST_SNAP    FIXED BIN(31),
              2 M(15)         LIKE S.MAXI,
              2 NO(15)        LIKE S.NO;
```

```
DCL   STAT FILE RECORD SEQUENTIAL INPUT
      ENVIRONMENT (F(1744) CONSECUTIVE);

DCL   F1 FILE PRINT;


ON ENDFILE (STAT) BEGIN;
      GOTO SUMMARY;
      END;


ON ENDFILE (SYSIN) BEGIN;
      CLCSE FILE (SYSIN);
      END;


ON ENDPAGE (F1) BEGIN;
      PUT FILE (F1) PAGE LINE (9);
      END;

OPEN FILE (F1) PAGESIZE (80);
```

```
FORM1:  PROCEDURE;
        /* PRINT STATISTICS FORM1 */

        PUT FILE (F1) PAGE LINE(9) EDIT
        ('INITIATORS','(NUMBER AND ASSOCIATED JOB CLASSES)',' ')
        (X(45),A(10),SKIP,X(33),A(35),SKIP(2),X(23),A(1));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
            PUT FILE (F1) EDIT (IN(I)) (X(5),F(2));
            END;

        PUT FILE (F1) SKIP EDIT (' ') (X(25),A(1));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
            PUT FILE (F1) EDIT (CLASSES(I)) (X(1),A(6));
            END;

        PUT FILE (F1) SKIP;

        DO J=1 TO 15;
            PUT FILE (F1) SKIP(2) EDIT
            ('CLASS ',SUBSTR(CLASS,J,1),'1:')
            (X(15),A(6),A(1),X(2),A(2));

            DO I=1 TO 7 WHILE (IN(I) ¬= 0);
                PUT FILE (F1) EDIT
                (S.JOB_START(I,J))(X(3),F(4));
                END;

            PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
            DO I=1 TO 7 WHILE (IN(I) ¬= 0);
                TIME = S.TIME_SJ(I) + S.TIME_WM(I) + S.TIME_WD(I)
                    + S.TIME_WS(I) + S.TIME_WV(I) + S.TIME_WW(I);
                IF TIME ¬= 0
                    THEN TIME = S.JOB_START(I,J) * 60 / TIME;
                PUT FILE (F1) EDIT
                (TIME) (X(2),F(5,3));
                END;
        END;

        PUT FILE (F1) SKIP(3) EDIT
        ('1:  NUMBER OF JOBS STARTED (TOTAL)'
        ,'2:  NUMBER OF JOBS STARTED (PER MIN.)')
        (X(24),A(34),SKIP,X(24),A(37));

        PUT FILE (F1) SKIP(5) EDIT ('TIME OF SNAPSHOT:',TIME_SNAP)
        (X(15),A(17),F(6));

        END FORM1;
```

```
FCRM2:  PROCEDURE;
        /* PRINT STATISTICS FORM2 */

        PUT FILE (F1) PAGE LINE(9) EDIT
        ('INITIATORS','(NUMBER AND ASSOCIATED JCB CLASSES)',' ')
        (X(45),A(10),SKIP,X(33),A(35),SKIP(2),X(23),A(1));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           PUT FILE (F1) EDIT (IN(I)) (F(7));
           END;

        PUT FILE (F1) SKIP EDIT (' ') (X(25),A(1));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           PUT FILE (F1) EDIT (CLASSES(I)) (X(1),A(6));
           END;

        PUT FILE (F1) SKIP(3) EDIT ('TIME_AC  1:') (X(15),A(11));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           SUM(I) = S.TIME_SJ(IN(I)) + S.TIME_WM(IN(I))
                  + S.TIME_WD(IN(I)) + S.TIME_WV(IN(I))
                  + S.TIME_WS(IN(I)) + S.TIME_WW(IN(I));
           PUT FILE (F1) EDIT (SUM(I),'  ') (F(6),A(1));
           END;

        PUT FILE (F1) SKIP(2) EDIT ('TIME_SJ  1:') (X(15),A(11));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           PUT FILE (F1) EDIT (S.TIME_SJ(IN(I)),'  ') (F(6),A(1));
           END;

        PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           PUT FILE (F1) EDIT (S.MAX_SJ(IN(I)),'  ') (F(6),A(1));
           END;

        PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           IF SUM(I) = 0 THEN TIME = 0;
           ELSE TIME = S.TIME_SJ(IN(I)) * 100 / SUM(I);
           PUT FILE (F1) EDIT (TIME,'  ') (F(6,1),A(1));
           END;

        PUT FILE (F1) SKIP(2) EDIT ('TIME_WM  1:') (X(15),A(11));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           PUT FILE (F1) EDIT (S.TIME_WM(IN(I)),'  ') (F(6),A(1));
           END;

        PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
        DO I=1 TO 7 WHILE (IN(I) ¬= 0);
           PUT FILE (F1) EDIT (S.MAX_WM(IN(I)),'  ') (F(6),A(1));
           END;
```

```
PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    IF SUM(I) = 0 THEN TIME = 0;
    ELSE TIME = S.TIME_WM(IN(I)) * 100 / SUM(I);
    PUT FILE (F1) EDIT (TIME,'  ') (F(6,1),A(1));
    END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WD  1:') (X(15),A(11));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    PUT FILE (F1) EDIT (S.TIME_WD(IN(I)),'  ') (F(6),A(1));
    END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    PUT FILE (F1) EDIT (S.MAX_WD(IN(I)),'  ') (F(6),A(1));
    END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    IF SUM(I) = 0 THEN TIME = 0;
    ELSE TIME = S.TIME_WD(IN(I)) * 100 / SUM(I);
    PUT FILE (F1) EDIT (TIME,'  ') (F(6,1),A(1));
    END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WV  1:') (X(15),A(11));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    PUT FILE (F1) EDIT (S.TIME_WV(IN(I)),'  ') (F(6),A(1));
    END;

PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    PUT FILE (F1) EDIT (S.MAX_WV(IN(I)),'  ') (F(6),A(1));
    END;

PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    IF SUM(I) = 0 THEN TIME = 0;
    ELSE TIME = S.TIME_WV(IN(I)) * 100 / SUM(I);
    PUT FILE (F1) EDIT (TIME,'  ') (F(6,1),A(1));
    END;

PUT FILE (F1) SKIP(2) EDIT ('TIME_WS  1:') (X(15),A(11));
DO I=1 TO 7 WHILE (IN(I) ¬= 0);
    PUT FILE (F1) EDIT (S.TIME_WS(IN(I)),'  ') (F(6),A(1));
    END;
```

```
PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));                          STA02120
DO I=1 TO 7 WHILE (IN(I) ¬= 0);                                       STA02130
     PUT FILE (F1) EDIT (S.MAX_WS(IN(I)),'  ') (F(6),A(1));           STA02140
     END;                                                             STA02150
                                                                      STA02160
PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));                          STA02170
DO I=1 TO 7 WHILE (IN(I) ¬= 0);                                       STA02180
     IF SUM(I) = 0 THEN TIME = 0;                                     STA02190
     ELSE TIME = S.TIME_WS(IN(I)) * 100 / SUM(I);                     STA02200
     PUT FILE (F1) EDIT (TIME,'  ') (F(6,1),A(1));                    STA02210
     END;                                                             STA02220
                                                                      STA02230
PUT FILE (F1) SKIP(2) EDIT ('TIME_WW  1:') (X(15),A(11));             STA02240
DO I=1 TO 7 WHILE (IN(I) ¬= 0);                                       STA02250
     PUT FILE (F1) EDIT (S.TIME_WW(IN(I)),'  ') (F(6),A(1));          STA02260
     END;                                                             STA02270
                                                                      STA02280
PUT FILE (F1) SKIP EDIT ('2:') (X(24),A(2));                          STA02290
DO I=1 TO 7 WHILE (IN(I) ¬= 0);                                       STA02300
     PUT FILE (F1) EDIT (S.MAX_WW(IN(I)),'  ') (F(6),A(1));           STA02310
     END;                                                             STA02320
                                                                      STA02330
PUT FILE (F1) SKIP EDIT ('3:') (X(24),A(2));                          STA02340
DO I=1 TO 7 WHILE (IN(I) ¬= 0);                                       STA02350
     IF SUM(I) = 0 THEN TIME = 0;                                     STA02360
     ELSE TIME = S.TIME_WW(IN(I)) * 100 / SUM(I);                     STA02370
     PUT FILE (F1) EDIT (TIME,'  ') (F(6,1),A(1));                    STA02380
     END;                                                             STA02390
                                                                      STA02400
PUT FILE (F1) SKIP(3) EDIT                                            STA02420
('TIME_AC: TIME ACTIVE  (=TIME_SJ + SUM OF ALL WAITING TIMES)'        STA02430
,'TIME_SJ: TIME SERVING JOBS (= ELAPSED JOB RUN TIME)'                STA02440
,'TIME_WM: TIME WAITING FOR MAIN MEMORY                      '        STA02450
,'TIME_WD: TIME WAITING FOR DEVICE(S)                        '        STA02460
,'TIME_WV: TIME WAITING FOR VOLUME(S) TO BE MOUNTED          '        STA02470
,'TIME_WS: TIME WAITING FOR DIRECT ACCESS SPACE              '        STA02480
,'TIME_WW: TIME WAITING FOR WORK                             '        STA02490
,'1:  TOTAL TIME IN SEC.                                     '        STA02500
,'2:  MAXIMUM TIME IN SEC.                                   '        STA02510
,'3:  TIME IN % OF ACTIVE TIME                               ')       STA02520
(X(15),A(59),SKIP,X(15),A(51),SKIP,X(15),A(51),SKIP,X(15),A(51)STA02530
,SKIP,X(15),A(51),SKIP,X(15),A(51),SKIP,X(15),A(51),SKIP(2)           STA02540
,X(15),A(51),SKIP,X(15),A(51),SKIP,X(15),A(51));                      STA02550
PUT FILE (F1) SKIP(5) EDIT                                            STA02570
('TIME OF SNAPSHOT:',TIME_SNAP) (X(15),A(17),F(6));                   STA02580
                                                                      STA02590
END FORM2;                                                            STA02600
```

```
FCRM3:  PRCCEDURE;
        /* PRINT STATISTICS FORM3 */

        PUT FILE (F1) PAGE LINE(9) EDIT
        ('J.A. #','J.S. #','J.S. %','MAX. W','MEAN W','DEV. W')
        (X(23),A(6),X(2),A(6),X(2),A(6),X(2),A(6),X(2),A(6),X(2),A(6));
        DO J=1 TO 15;
              JOBS_AVAIL = S.JOB_IN(J) + S.IN_QUEUE(J);
              START = 0;
              DO I=1 TC 15;
                    START = START + S.JOB_START(I,J);
                    END;
              PUT FILE (F1) SKIP(2) EDIT
              ('CLASS ',SUBSTR(CLASS,J,1),JOBS_AVAIL,START)
              (X(15),A(6),A(1),F(6),F(8));
              IF START = 0 THEN PUT FILE (F1) EDIT
                    ('--','--','--') (X(6),A(2),X(6),A(2),X(6),A(2));
              ELSE DO;
                    AVERAGE = S.JOB_WAIT(J) / START;
                    IF S.JOB_IN(J) = 0 THEN PERCENT = 0;
                    ELSE PERCENT = START * 100 / JOBS_AVAIL;
                    PUT FILE (F1) EDIT
                    (PERCENT,S.JOB_WAITM(J),AVERAGE) (F(8,1),2F(8));
                    END;
              IF START < 2 THEN PUT FILE (F1) EDIT ('--') (X(6),A(2));
              ELSE DO;
                    VARIANCE = (S.JOB_WAITS(J) - START * AVERAGE ** 2)
                               / (START - 1);
                    DEVIATION = SQRT(VARIANCE);
                    PUT FILE (F1) EDIT (DEVIATION) (F(8));
                    END;
              END;
        PUT FILE (F1) SKIP(3) EDIT
        ('J.A. # = JOBS AVAILABLE (TOTAL NUMBER)                          '
        ,'J.S. # = JOBS STARTED (TOTAL NUMBER)                            '
        ,'J.S. % = JOBS STARTED (IN % OF JOBS AVAILABLE)                  '
        ,'MAX. W = MAX. WAITING TIME PER JOB TO GET STARTED (IN SEC)'
        ,'MEAN W = MEAN WAITING TIME PER JOB TO GET STARTED (IN SEC)'
        ,'DEV. W = STANDARD DEVIATION OF WAITING TIME                     ')
        (X(15),A(58),SKIP,X(15),A(58),SKIP,X(15),A(58),SKIP,X(15)
        ,A(58),SKIP,X(15),A(58),SKIP,X(15),A(58));
        PUT FILE (F1) SKIP(5) EDIT ('TIME OF SNAPSHOT:',TIME_SNAP)
        (X(15),A(17),F(6));

        END FORM3;
```

```
PREPARE:    PROCEDURE;
         /* PREPARE SUMMERY */

         CURRENT_SNAP    = TIME_SNAP;
         TOTAL           = TOTAL + S, BY NAME;
         TOTAL.IN_QUEUE  = 0;
         N               = N + 1;
         IF N=1 THEN FIRST_SNAP = TIME_SNAP;
         DO I=1 TO 15;
            M.JOB_WAITM(I) = MAX(M.JOB_WAITM(I),S.JOB_WAITM(I));
            END;

         DO I=1 TO 7 WHILE (IN(I) ¬= 0);
            M.MAX_SJ(IN(I)) = MAX (M.MAX_SJ(IN(I)), S.MAX_SJ(IN(I)));
            M.MAX_WD(IN(I)) = MAX (M.MAX_WD(IN(I)), S.MAX_WD(IN(I)));
            M.MAX_WM(IN(I)) = MAX (M.MAX_WM(IN(I)), S.MAX_WM(IN(I)));
            M.MAX_WS(IN(I)) = MAX (M.MAX_WS(IN(I)), S.MAX_WS(IN(I)));
            M.MAX_WV(IN(I)) = MAX (M.MAX_WV(IN(I)), S.MAX_WV(IN(I)));
            M.MAX_WW(IN(I)) = MAX (M.MAX_WW(IN(I)), S.MAX_WW(IN(I)));
            END;

         END PREPARE;
```

```
        TOTAL = 0;
        N     = 0;
        PUT LIST ('ENTER INITIATORS:');
        PUT SKIP;
        GET LIST (IN(1), IN(2), IN(3), IN(4), IN(5), IN(6), IN(7));


LOOP:
        READ FILE (STAT) INTO (S);
        PUT LIST ('ENTER STATISTICS TYPES:');
        PUT SKIP;
        GET LIST (T(1), T(2), T(3), T(4));

        IF T(4) = 1 THEN CALL PREPARE;
        IF T(1) = 1 THEN CALL FORM1;
        IF T(2) = 1 THEN CALL FORM2;
        IF T(3) = 1 THEN CALL FORM3;
        GOTO LOOP;

SUMMARY:
        PUT SKIP LIST ('ENTER SUMMARY TYPES:');
        PUT SKIP;
        GET LIST (T(1), T(2), T(3));

        TIME_SNAP = CURRENT_SNAP;
        S = TOTAL, BY NAME;
        MAXI = M;
        IF T(1) = 1 THEN CALL FORM1;
        IF T(2) = 1 THEN CALL FORM2;
        IF T(3) = 1 THEN CALL FORM3;

        END STA;
```

BIBLIOGRAPHY

1. Afifi, A.A. and Azen, S.P., Statistical Analysis, A Computer Oriented Approach, Academic Press, 1972.

2. Barron, D.W., Computer Operating Systems, Chapman and Hall Ltd., 1971.

3. Browne, J.C., Lan, J. and Baskett, F., "The Interaction of Multiprogramming Job Scheduling and CPU Scheduling", Proceedings, AFIPS, FJCC, vol. 41, part 1, p. 13-22, 1972.

4. Colin, A.J.T., Introduction to Operating Systems, American Elsevier Publishing Co., 1971.

5. Crowly J.J.Jr., Lt USN, and Karns, L.N., Lt USN, An Interactive Simulation of the IBM System 360 Operator's Console, M.S. Thesis, Naval Postgraduate School, Monterey, California 93940, 1976.

6. Cuttle, G. and Robinson, P.B., Executive Programs and Operating Systems, American Elsevier Publishing Co., 1970.

7. Flores, I., Computer Programming System/360, Prentice-Hall, 1971.

8. Flores, I., Job Control Language and File Definition, Prentice-Hall, 1971.

9. Flores, I., OS/MVT, Prentice-Hall, 1973.

10. Freeman, P., Software Systems Principles, Science Research Associates, 1975.

11. Hoare, C.A.R. and Perrot, R.H., Operating Systems Techniques, Academic Press, 1972.

12. IBM System/360 Operating System: Introduction, 5th ed., IBM, 1972.

13. IBM System/360 Operating System: MVT Guide, 6th ed., IBM, 1974.

14. IBM System/360 Operating System: MVT Job Management, Program Logic Manual, 10th ed., IBM, 1971.

15. IBM System/360 Operating System: MVT Supervisor, 7th ed., IBM, 1972.

16. IBM System/360 Operating System: Operator's Reference, OS Release 21.7, 4th ed., IBM, 1974.

17. Katzan, H., Jr., Computer Organization and the System/370, Van Nostrand Reinhold, 1971.

18. Katzan, H., Jr., Operating Systems, Van Nostrand Reinhold, 1973.

19. Katzan, H., Jr., Information Technology, Petrocelli Books, 1974.

20. Madnick, S.E. and Donavan, J.J., Operating Systems, McGraw-Hill Book Co., 1974.

21. Sayers, A.P. , editor, Operating Systems Survey, Auerbach Publishers, 1971.

22. User's Manual, W.R. Curch Computer Center, 2nd ed., Naval Postgraduate School, Monterey, California, 1974.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, Virginia 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, California 93940	2
3.	Department Chairman, Code 52 Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
4.	Professor N. F. Schneidewind, Code 52SS Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
5.	Professor R. J. Roland, Lt.Col., USAF, Code 52RL Department of Computer Science Naval Postgraduate School Monterey, California 93940	1
6.	Professor D. J. Williams Director W. R. Church Computer Center, Code 0141 Naval Postgraduate School Monterey, California 93940	1

7. Mr. D. F. Norman 1
Manager Operations Group, Code 0141
Naval Postgraduate School
Monterey, California 93940

8. Marineamt -A1- 1
294 Wilhelmshaven
Federal Republic of Germany

9. Dokumentationszentrale der Bundeswehr (See) 1
53 Bonn
Friedrich-Ebert-Allee 34
Federal Republic of Germany

10. BMVg - P V 13 - 1
53 Bonn
Postfach
Federal Republic of Germany

11. KptLt Erik Fiegl 1
Marinefernmeldegruppe 11
2392 Gluecksburg-Meierwik
Federal Republic of Germany

Thesis 170357
F395 Fiegl
c.1 A simulation model
for the study of job
scheduling policy.